ABYSS ETH2 DEPOSITOR SMART CONTRACT AUDIT

February 24, 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 Abyss Finance. 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

Smart contract that allows convenient way to send 1 to 100 deposits in one transaction to Eth2 Deposit Contract.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 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 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> 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

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

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> 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

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> 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
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 includes smart contract that allows users to stake their Ethereum assets to Ethereum v2 deposit contract.

1.5 PROJECT DASHBOARD

Client	Abyss Finance
Audit name	Abyss Eth2 Depositor
Initial version	a2d58dea4d79846dc682fe93ac3e0eca02323d11
Final version	179ff88f713885aca2aca5defb5e4e0a74ee02f1
SLOC	68
Date	2021-02-02 - 2021-02-24
Auditors engaged	2 auditors

FILES LISTING

AbyssEth2Depositor.sol	AbyssEth2Depositor.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	2
Comment	1

CONCLUSION

Smart contracts have been audited and no critical or major issues were found, several recommendations were marked as warning and comment. During work on audit report all issues were fixed or acknowledged by client, so contracts assumed as secure to use according our security criteria. Final commit identifier with all fixes: `179ff88f713885aca2aca5defb5e4e0a74ee02f1`. Deployed contract address: `0xFA5f9EAa65FFb2A75de092eB7f3fc84FC86B5b18`.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Absent ETH2 depositor BLS signature parameters check
File	AbyssEth2Depositor.sol
Severity	Warning
Status	Fixed at a36365f6

DESCRIPTION

This warning is about the function `deposit` in here: AbyssEth2Depositor.sol#L61 having inputs for a pretty complicated cryptographic primitive (BLS12-381 with switched G1 and G2 groups inside) with no checks for their correctness at all.

This can result in faulty signatures, public keys and withdrawal credentials to be submitted which will lead to the loss of withdrawal possibility later.

RECOMMENDATION

It is recommended to implement following checks (according to Ethereum usage of variant 2 of: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bls-signature-04#section-2.1):

- Signature length to be equal to 96 bytes.
- Public key length to be equal to 48 bytes.
- Withdrawal credentials, in case they are supposed to be aggregatable public keys, each of them should be equal to 32 bytes.

WRN-2	Reentrant ETH2 depositor function
File	AbyssEth2Depositor.sol
Severity	Warning
Status	Fixed at a36365f6

DESCRIPTION

This warning is about the function `deposit` (AbyssEth2Depositor.sol#L61) being possibly reentrant in here: AbyssEth2Depositor.sol#L80.

This can lead to unpredictable deposits chain initially not supposed to be done by the application logic.

RECOMMENDATION

It is recommended to introduce `nonReentrant` modifier, bringing it from (probably) OpenZeppelin's implementation in here: ReentrancyGuard.sol.

2.4 COMMENTS

CMT-1	ETH2 depositor argument design consideration
File	AbyssEth2Depositor.sol
Severity	Comment
Status	No issue

DESCRIPTION

This comment is about absence of possibility to check the correctness of the SHA2-256 hash being passed as a parameter `deposit_data_roots` to the `deposit` function in here: AbyssEth2Depositor.sol#L61.

RECOMMENDATION

Since the internal structure of data being hashed makes sense, it is better to use some ZK-proof (with Zookrates possibly?), which will bring the possibility to verify the correctness of the input data without having the actual data present.

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