ETHA Lend Code Audit

By

Footprints Tech

Table of Contents

1 INTRODUCTION

1.1. DOCUMENT PURPOSE

The findings of the review are presented in this document.

1.2. REVIEW GOALS & FOCUS

Sound Architecture

This review includes both objective findings from the contract code as well as subjective assessments of the overall architecture and design choices. Given the subjective nature of certain findings commentary from the Etha Protocol development team has been included in the report as appropriate.

Smart Contract Best Practices

This review will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This review will evaluate whether the code does what it is intended to do.

Code Quality

This review will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This review will look for exploitable security vulnerabilities.

1.3. TERMINOLOGY

This review uses the following terminology:

Severity Terms

Measure the magnitude of an issue.

Minor

Minor issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Minor issues in general will not indicate an actual problem or bug in code. The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable such as requiring a specific condition to arise to be exploited. Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

Critical

Critical issues are directly exploitable bugs or security vulnerabilities. Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.

2. SOURCE CODE

2.1. REPOSITORY

This audit has been done for contracts in the following repository:

https://github.com/etharemit/etha-protocol/tree/development/contracts

2.2. CONTRACTS

The code audit carried out was limited to the folder containing smart contract(s) shared over the repository as below: -

1. **Wallet**
2. **Registry**
3. **Logic**
    1. Dydx
    2. Aave
    3. Compund
    4. Uniswap
    5. Curve
4. **Adapters**

3. DISCLAIMER

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The Audit also holds no responsibility on the analysis given by open source tools used in this report. The output of the tools should only be used as help in the analysis.

4. FINDINGS

This section contains detailed issues and analysis.

4.1 CRITICAL ISSUES

Issue#: 001 No zero address check

Contract Name#: EthaRegistry.sol

Contract Location#: /contracts/registry/EthaRegistry.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **withdraw(address erc20,address recipient,uint256 amount)**, there is no zero address check for "erc20", "recipient". If recipient is zero address then we may be sending the amount to zero address which results in loss of amount.

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 002 Underflow Issue while subtraction

Contract Name#: AaveLogic.sol

Contract Location#: /contracts/logics/AaveLogic.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **setApproval(address erc20, uint256 srcAmt, address to)**, safemath should be used for subtraction at line**#:**224 to prevent underflow. Underflow can be understood as, suppose you're using an unsigned integer in Solidiy. The possible values of your variable ranges from 0 to 2^256 (1.1579209e+77). So that if you're around the minimum value 0 and decrement your variable it will go back to 2^256.

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 003 Underflow Issue while subtraction

Contract Name#: CompoundLogic.sol

Contract Location#: /contracts/logics/CompundLogic.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **repayToken(address erc20, address cErc20, uint256 tokenAmt)**, safemath should be used for subtraction at line**#:**388 and line **#:**351 to prevent underflow. Underflow can be understood as, suppose you're using an unsigned integer in Solidiy. The possible values of your variable ranges from 0 to 2^256 (1.1579209e+77). So that if you're around the minimum value 0 and decrement your variable it will go back to 2^256.

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 004 Overflow Issue while addition

Contract Name#: CurveLogic.sol

Contract Location#: /contracts/logics/CurveLogic.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **swapOnCurveCompound(address src, address dest, uint256 srcAmt)**, safemath should be used for addition at line**#:**134,line**#:**135, line**#:**151, line**#:**152, line**#:**154, line**#:**155, line**#:**171, line**#:**172, line**#:**173, line**#:**175, line**#:**176, line**#:**177, line**#:**193, line**#:**194, line**#:**195, line**#:**197, line**#:**198, line**#:**199, line**#:**215, line**#:**216, line**#:**217, line**#:**219, line**#:**220, and line**#:**221 to prevent overflow. Overflow can be understood as, suppose you're using an unsigned integer in Solidiy. The possible values of your variable ranges from 0 to 2^256 (1.1579209e+77). So that if you're around the maximum value value 2^256 and increment your variable it will go back to 0.

Action: Acknowledged. The response from the Etha Lend team regarding this issue is, "SafeMath is not required here because the value would be either 1, 2, 3, or 4 only.

4.2 MAJOR ISSUES

Issue#: 001 No zero address check

Contract Name#: SmartWallet.sol

Contract Location#: /contracts/wallet/SmartWallet.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **initialize(address _registry, address _user)**, there is no zero address check for "_user".

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 002 No zero address check

Contract Name#: EthaRegistry.sol

Contract Location#: /contracts/registry/EthaRegistry.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **enableLogic(address _logicAddress)**, there is no zero address check for "_logicAddress"

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 003 No zero address check

Contract Name#: EthaRegistry.sol

Contract Location#: /contracts/registry/EthaRegistry.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **disableLogic(address _logicAddress)**, there is no zero address check for "_logicAddress"v

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 004 No zero address check

Contract Name#: EthaRegistry.sol

Contract Location#: /contracts/registry/EthaRegistry.sol

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In function **initialize(address _owner,address _feeRecipient,uint256 _fee)**, there is no zero address check for "_owner", "_feeRecipient"

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

Issue#: 005 No zero address check

Contract Name#: All logic contract including dydx, aave, compound

Contract Location#: /contracts/logics

Codes commit#: 1646b6fbda98f0299fb98506d908505a52ddedee

Description:

In most of the functions there is no zero address check for "_owner", "_feeRecipient"

Action: Fixed the issue at Commit# **96f272821a4c9c9b1b817f658e87b00caf678074**

4.3 MINOR ISSUES

Outdated Solidity version: The latest solidity version at the time of this writing is 0.7.4. The contract uses version 0.5.*.

Action: Acknowledged. Didn't changed the versions because thorough testing had been done on current version.

5. GENERAL COMMENTS

Following are some general comments which should be fixed/modified for better code quality: -

- The contracts can be optimized by using require and assert at their proper places. Here is a link to study about it, https://ethereum.stackexchange.com/questions/15166/difference-between-require-and-assert-and-the-difference-between-revert-and-thro
- The contract is coded to be compiled with a solidity version greater than ^0.5.0 or ^0.5.2. It is recommended to remove the cap ^ before 0 in above version because it may get compiled with any of the versions above version 5. The problem with this is if any breaking change comes in any of the new versions, the code would fail there.

  Additionally, it is advised to write the code in the newer version of solidity Few improvements and new features have been added in this solidity version.

  Below is one of them-

  Fallback function is split into following two types of function –

  - receive – invoked if ether is sent to a contract

- fallback – invoked if data is sent to a contract or if ether is sent to a contract(when receive is not defined)