Smart Contract Lite Audit Report for Aries Finance

AnChain.AI Inc.

Auditors: AnChain.AI Audit Team

Contact: Audit@AnChain.ai

Date: 2021-02-19

Index

1. Introduction

1.1 Audit Info

This document includes the results of the security lite audit for the client's smart contract code as found in the section 3. The security audit was conducted by the AnChain.AI team from Feb 09, 2021, to Feb 19, 2021.

The purpose of this audit is to review source code, functionality on test net, and provide feedback on the design, architecture, and quality of the source code with an emphasis on validating the correctness and security of the software in its entirety.

1.2 Methodology

AnChain.AI team adopts a suite of tools and best practice from cybersecurity to audit smart contract source code for vulnerabilities.

3 key aspects in the AnChain.AI security audit methodology:

- Vulnerability audit. We use various tools to scan for known vulnerabilities, including our proprietary CAS sandbox.
- Statistical audit. Our sandbox will predict the risk score for the audited code with hundreds of thousands of mainnet smart contracts to understand security stance.
- Business logic audit. Our experts will design specific test cases to cover the key custom functions, to identify potential risk exposures.

1.3 Environment

Compiled Version: Solidity Compiler v0.7.5+commit.eb77ed08

Compiled System: macOS

Compiled Tool: Remix IDE, Truffle

2. Executive Summary

2.1 Key Findings

We **did not identify high severity vulnerabilities** that would compromise the integrity of the project. This smart contract **meets** the AnChain.AI smart contract lite auditing standards.

3. Smart Contract Scope

3.1 Audited Source Code File

File Name	Md5
AFI.sol	`fefa45dce725666007beb405858a5bf2`
Controller.sol	`bc104dbb5941b85e7c3090236fb34a0d`
afiVault(USDC).sol	`531092DBDAC02D178E1B4CA6B9260572`
afiVault(USDT).sol	`DEFB0FA72312B85C4B542A63925D031E`
afiVault(DAI).sol	`21B8B8AFF2EF5DC23A8507A8250215D4`
rewards(Uni_V2).sol	`07F49AC886E941DFE331B98FA319ED99`
rewards(AFI).sol	`07F49AC886E941DFE331B98FA319ED99`
rewards(afiUSDC).sol	`07F49AC886E941DFE331B98FA319ED99`
rewards(afiUSDT).sol	`07F49AC886E941DFE331B98FA319ED99`
rewards(afiDAI).sol	`07F49AC886E941DFE331B98FA319ED99`
StrategyDForceUSDT.sol	`E917E24029D8590C4BBB4B7B9EA75B31`
StrategyDForceUSDC.sol	`73F9A4D1E90769D8B1DF4AFB544943B3`
StrategyDForceDAI.sol	`9E6C3834B60E88D240BDA2CE581C22F2`
BonusRewards.sol	`06432D38ABC21A28DC6C93E828FD1409`

4. Code Difference

This project is originated from YFI[1]. After comparing the code difference, we identified that the project team made a few changes without affecting the original project feature. Here is a table containing statistics of code difference:

File Name	Lines of Removals	Lines of Additions	Number of Functions Edited	Comment
AFI.sol	0	0	0	
Controller.sol	8	5	0	
afiVault(USDC).sol	131	54	4	Removed ***depositAll()***, ***withdrawAll()***, ***harvest()*** and edited ***deposit()***
afiVault(USDT).sol	3	7	0	
afiVault(DAI).sol	3	7	0	
rewards(Uni_V2).sol	139	212	3	Added ***constructor***, ***getData1()***, ***getData2()***
rewards(AFI).sol	139	212	3	Added ***constructor***, ***getData1()***, ***getData2()***
rewards(afiUSDC).sol	139	212	3	Added ***constructor***, ***getData1()***, ***getData2()***
rewards(afiUSDT).sol	139	212	3	Added ***constructor***, ***getData1()***, ***getData2()***
rewards(afiDAI).sol	139	212	3	Added ***constructor***, ***getData1()***, ***getData2()***
StrategyDForceUSDT.sol	52	224	0	
StrategyDForceUSDC.sol	85	253	1	Removed ***getName()***
StrategyDForceDAI.sol	84	240	2	Removed ***getName()*** and edited ***withdraw()***
BonusRewards.sol	236	229	8	Removed original ***YearnRewards*** contract and added ***BonusReward*** contract for airdrop

The detailed difference can be found in attachment, with original YFI project on the left side and this project on the right side.

5. Original Project Audit

The original project YFI was audited by Quantstamp and the report[2] stated that no significant code-related issue is found. This project is secured without making major changes to the original project YFI.

6. Conclusion

This project has the basic logic of YFI, with additional editing on view functions without changing the original project design and additional airdrop contract. We have verified this revision does not affect security and normal usage of the smart contract. Hence we confirmed that this project is secured.

7. Appendix

7.1 Vulnerability Technical Explanation

These vulnerability scannings are powered by the patented AnChain.AI CAS (Smart Contract Auditing Sandbox), including static analysis, dynamic analysis, and statistical analysis.

- **Numerical Overflow**

Severity Level: high

In ETH, failing to run a numerical check on arithmetic operations may cause the values to overflow or underflow, which may cause crypto asset loss. It is a good practice to use "SafeMath" library for all the numeric operations to take care of overflow and underflow issue of all the numeric operations.

The code provided pass the 'numerical overflow' vulnerability check.

- **Authorization**

Severity Level: high

Parameters passed into a function should be consistent with the actual caller.

The best practice is to use 'require()' to check if the user executing the action matches the provided parameter.

The code provided pass the 'Authorization' vulnerability check.

- **Transaction Ordering Dependency:**

Severity Level: high

In ETH, timestamp of a transaction getting approved can be manipulated by vicious minors. Transaction Ordering Dependency refer to critical logic fault if minors approve the later submitted transaction PRIOR than the earlier ones.

Transaction ordering dependency does not have critical impact on this contract.

- **Parity Multisig Bug/ Delegate Call:**

Severity Level: High

The code does not call any external contracts and thus there is no issue about parity multisig bug.

- **Random Number**

Severity Level: High

Random number generator algorithm should not use predictable seeds.

The code does not generate or use any random number, and thus there is no issue about the random number issue.

The code provided pass the "Random Number" vulnerability check.

- **Re-entrancy Attack:**

Severity Level: High

The code does not contain any function that is vulnerable to re-entrancy attack.

- **Function Visibility:**

Severity Level: Low

The code does not contain any function that is improperly set with visibility.

- **Event Emitting:**

Severity Level: Low

The ERC code has emit transfer & approval event properly, which meets a ERC token standard.

- **ERC Token Standard:**

Severity Level: Low

The token contract has the following variables: name, symbol, decimal, totalSupply, which meets the ERC token standard.

7.2 Severity Level

Level	Description
High	The issue poses an existential risk to the project, and the issue identified could lead to massive financial or reputational repercussions.
Medium	The potential risk is large, but there is some ambiguity surrounding whether or not the issue would practically manifest.
Low	The risk is small, unlikely, or not relevant to the project in a meaningful way.
Code Quality	The issue identified does not pose any obvious risk, but fixing it would improve overall code quality, conform to recommended best practices, and perhaps lead to fewer development issues in the future.

7.3 Reference

[1]. YFI Source Code. Retrieved from https://github.com/iearn-finance

[2]. YFI Audit Report. Retrieved from https://quantstamp.com/blog/yearn-finance-security-review

[3]. YFI Audit Repository. Retrieved from https://github.com/iearn-finance/yearn-security/tree/master/audits

8. Disclaimer

Disclaimer - Smart Contract Auditing - ETH

AnChain.ai makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and AnChain.ai specifically disclaims all implied warranties or merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

AnChain.ai will not be liable for any lost profits, business, contracts, revenues, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand against the company by any other party. If no event will AnChain.ai be liable for consequential, incidental, special, indirect, or exemplary damages arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if AnChain.ai has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the client's team and only the source code AnChain.ai notes as being within the scope of AnChain.ai's review within this report. This report does not include an audit of the deployment scripts used to deploy the contracts in the repository corresponding to this audit. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. In this report you may through hypertext or other computer links, gain access to websites operated by persons other than AnChain.ai. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' owners. You agree that AnChain.ai is not responsible for the content or operation of such websites and that AnChain.ai shall have no liability to your or any other person or entity for the use of third party websites. AnChain.ai assumes no responsibility for the use of third-party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.