Vigilantia Operational

ARK Finance
Security Audit

Website: https://www.arkfi.io/
Date of completion: 26/02/2023

Request an audit
Email: vigilantiaoperational@proton.me
Twitter: https://twitter.com/VigilantiaWeb3

1. Disclaimer

Security audits cannot uncover all existing vulnerabilities; even an audit in which no vulnerabilities are found is not a guarantee of a secure system.

However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it.

Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code audits aimed at determining all locations that need to be fixed. Within the customer-determined time frame, we performed an audit in order to discover as many vulnerabilities as possible.

The focus of our audit was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself.

Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

2. About Ark Finance

Ark Fi is an innovative fintech group focused on creating simplified decentralized finance (DeFi) products for investors. Ark Fi aims to increase the adoption of DeFi protocols by creating simple to use applications community members are proud to introduce to their family, friends, neighbors and colleagues. The Ark Fi ecosystem of high yield applications are designed to be disconnected from the volatility of traditional markets and provide a passive income source during times when investors need it most. We understand crypto markets are inherently volatile and have designed the tokenomics and mechanics of Ark Fi products to be as stable as possible while creating an opportunity for high rewards. Through team member relationships and experience in cryptocurrency, trading, investing, business development and traditional finance, we aim to bring real-world investment revenue to our DeFi ecosystem, increasing the sustainability and long-term earning potential for our community of investors.

Documentation: https://docs.arkfi.io/

3. Smart Contract In Scope

ARK_LEGACY
https://bscscan.com/address/0x2222223B05B5842c918a868928F57cD3A0332222

Solidity Version:
v0.8.16+commit.07a7930e

Optimization Enabled:
Yes with 200 runs

SLOC (Source lines of code):
786

4. Terminology & Risk Classification

For the purpose of this audit, we adopt the following terminology: To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

Likelihood: The likelihood of a finding to be triggered or exploited in practice.

Impact: The technical and business-related consequences of a finding.

Severity: An additive consideration based on the Likelihood and the Impact (as referenced below).

We use the table below to determine the severity of any and all findings. Please note that none of these risk classifications constitute endorsement or opposition to a particular project or contract.

	Impact		
Likelihood	**High**	**Medium**	**Low**
High	Severity: Critical	Severity: High	Severity: Medium
Medium	Severity: High	Severity: Medium	Severity: Low
Low	Severity: Medium	Severity: Low	Severity: Low

4. Summary

In the period 20.02.2023 - 26.02.2023 we audited ARK Finance's ARK_LEGACY smart contract.

In this period of time a total of [number] of issues were found.

Severity	Issues
Critical Risk	-
High Risk	-
Medium Risk	1
Low Risk	1
Gas Optimization	3
Informational	-

5. Issues

5.1 [M - 001]: Function „airdropEveryone()" might not be executed if the „minted" variable is too high

Code:

```
    function airdropEveryone() external onlyCEO {
        uint256 minted = IERC721Enumerable(address(OLD)).totalSupply();
        address owner;
        uint256 level;
        uint256 sharesOfId;
        uint256 exludedRewards;
        uint256 unclaimedRewards;
        uint256 oldTotalRewardsPerShare = OLD.totalRewardsPerShare();
        for(uint256 i = 0; i < minted; i++){
            owner = OLD.ownerOf(i);
            level = OLD.levelOfNft(i);
            sharesOfId = sharesOfLevel[level];
            exludedRewards = OLD.excluded(i);
            mintToWallet(owner,level);
            locked[i] = true;
            excluded[i] = exludedRewards;
            unclaimedRewards += sharesOfId * (oldTotalRewardsPerShare -
            exludedRewards) / veryBigNumber;
            claimedRewards[i] = OLD.claimedRewards(i);
        }
        totalRewardsPerShare = oldTotalRewardsPerShare;
        uint256 busdAmount = OLD.rewardsPool() + unclaimedRewards;
        require(BUSD.transferFrom(msg.sender, address(this),
         busdAmount),"BUSD transfer failed");
        rewardsPool += OLD.rewardsPool();
        emit MigrationFinished(BUSD.balanceOf(address(OLD)),
rewardsPool, unclaimedRewards);
    }
```

Issue: Transaction might exceed the block's gas limit and fail.

Impact: If the „minted" variable (= IERC721Enumerable(address(OLD)).totalSupply()) is too high the transaction might exceed the block's gas limit and fail.

Mitigation: Use batch transfers where the caller input manually an array of users to airdrop instead of airdropping to everyone at once;

5.2 [L - 001]: Redundant code in „vote” function

Code:

```
    function vote(uint256 index, bool yourVote) external {
        require(votes[index].endTime >= block.timestamp, "Vote is
closed”);

        if(balanceOf(msg.sender) == 0) return;

        uint256 id = tokenOfOwnerByIndex(msg.sender, 0);
        uint256 level = levelOfNft[id];

        uint256 votesOfThisNft = votesOfLevel[level];

        if(votesOfThisNft == 0) return;
        if(voteCast[id][index]) {
            if(decisionOnVote[id][index]) votes[index].yes -=
      votesOfThisNft;

            if(!decisionOnVote[id][index]) votes[index].no -=
      votesOfThisNft;
        }

        if(yourVote) votes[index].yes += votesOfThisNft;
        if(!yourVote) votes[index].no += votesOfThisNft;

        decisionOnVote[id][index] = yourVote;
        voteCast[id][index] = true;
    }
```

Issue: Sequence of code without effect.

POC: If there are 3 users who voted with YES (bool yourVote = TRUE) using their NFTs (a, b, c), in a proposal, decisionOnVote[id][index] and decisionOnVote[id][index]voteCast[id] [index] are marked as True. If one of the users who voted call the function for the second time, the if statement „if(voteCast[id][index])” will be executed and the votes of the caller will be un-casted and casted again in the votes[index].yes variable.

Mitigation: Remove the if statement „if(voteCast[id][index])” and add a mapping variable that will allow users to vote once per proposal.

5.3 [G - 001]: Repetitive code found

Description:
(1)„function mint(uint256 level) external", "function mintToWallet(address to, uint256 level) public onlyCEO", "function mintToWalletPaid(address to, uint256 level) external onlyCEO" are using similar lines of code. Wrap them up in an internal function to reduce the gas consumption.

(2)„function distributeRewards() external onlyArk", "function distributeRewardsManuallyCEO() external onlyCEO" are using similar lines of code. Wrap them up in an internal function to reduce the gas consumption.

5.4 [G - 002]: Require statement found at the end of the function

Description:
"function mint(uint256 level) external", "function mintToWalletPaid(address to, uint256 level) external", "function levelUp(uint256 id) external", "function _claim(address investor) internal", "function airdropEveryone() external" are using the require statement „require(BUSD.transfer(CEO, price - rewardsShare),"BUSD transfer failed");" at the end of the function. In order to save gas, move it upward, at the beginning of the function.

5.4 [G - 003]: Math operations can be marked with „unchecked"

Description: Math operations from the following functions "function mint(uint256 level) external", "function mintToWallet(address to, uint256 level) public", "function mintToWalletPaid(address to, uint256 level) external", "function levelUp(uint256 id) external" can be marked as "UNCHECKED" in order to save more gas.