Audit Whale

Smart contract code review and security analysis report

This document may contain confidential information about IT systems and the intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for AnyStake.
Type	Staking protocol
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Approved by	Jameson| COO Audit Whale
Github link	https://github.com/defiat-crypto/anystake-contracts/tree/main/contracts
Commit for initial audit	d67d526121a2d24ed0f0b616bf6d81af78645484
Commit for secondary audit	4e91e27710af93ea096e682c0c0fd67fc7d1fb39
Timeline	3RD MAR 2021 – 1ST APR 2021
Changelog	11TH MAR 2021- Initial Audit 1ST APR 2021- Secondary Audit

Table of contents

Introduction 4
Scope 4
Executive Summary 5
Severity Definitions 6
AS-IS overview 7
Conclusion 7
Disclaimers 19

Introduction

Audit Whale (Consultant) was contracted by AnyStake (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between March 3rd, 2021 – March 11th, 2021; secondary review was conducted between March 29th, 2021 – April 1st, 2021.

Scope

The scope of the project is main net smart contract that can be found on github: https://github.com/defiat-crypto/anystake-contracts/tree/main/contracts
We have scanned this smart contract for commonly known and more specific vulnerabilities. List of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are secure.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section and all found issues can be found in the Audit overview section.

Security engineers found 3 critical, 3 high, 3 medium and 2 low severity issues during the initial audit. All the issues were fixed for the secondary audit.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets lose or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets lose or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

AnyStake smart contracts

AnyStake smart contract consists of AnyStake contract, AnyStakeRegulator contract, AnyStakeVault; and libs and contracts in interfaces, lib, mock, utils folders.

Interfaces

Description

IAnyStake, IAnyStakeMigrator, IAnyStakeRegulator, IAnyStakeVault, IDeFiatGov, IDeFiatPoints, IDeFiatToken, IUniswapV2Factory, IUniswapV2Pair, IUniswapV2Router01, IUniswapV2Router02, IWETH, IERC20 are interfaces for contracts interactions.

Standard libraries

Description

Ownable, Address, Context, SafeERC20, SafeMath are standard OpenZeppelin libraries and contracts for access control, math operations, context etc.

Utils

Description

DeFiatGovernedUtils, DeFiatUtils are helper smart contracts for the DeFiat platform.

AnyStakeUtils is a helper smart contract for AnyStake.

AnyStakeRegulator

Description

AnyStakeRegulator is a smart contract for DFTP staking.

Imports

AnyStakeRegulator has following imports:

- IDeFiatPoints.sol

- IAnyStakeMigrator.sol
- IAnyStakeRegulator.sol
- IAnyStakeVault.sol
- AnyStakeUtils.sol

Inheritance

AnyStakeRegulator contract is IAnyStakeRegulator, AnyStakeUtils.

Usages

AnyStakeRegulator contract has following usages:

- using SafeMath for uint256;

Structs

AnyStakeRegulator contract has following structs:

- struct UserInfo

  uint256 amount;

  uint256 rewardDebt;

  uint256 lastRewardBlock;

Enums

AnyStakeRegulator contract has no custom enums.

Events

AnyStakeRegulator contract has following events:

- event Initialized(address indexed user, address vault);
- event Claim(address indexed user, uint256 amount);
- event Deposit(address indexed user, uint256 amount);
- event Withdraw(address indexed user, uint256 amount);
- event Migrate(address indexed user, uint256 amount);
- event EmergencyWithdraw(address indexed user, uint256 amount);
- event StakingFeeUpdated(address indexed user, uint256 stakingFee);
- event BuybackRateUpdated(address indexed user, uint256 buybackRate);
- event PriceMultiplierUpdated(address indexed user, uint256 amount);
- event MigratorUpdated(address indexed user, address migrator);
- event VaultUpdated(address indexed user, address vault);
- event RegulatorActive(address indexed user, bool active);

Modifiers

AnyStakeRegulator contract has following modifiers:

- NoReentrant
- onlyVault
- activated

Fields

AnyStakeRegulator contract has following parameters:

- mapping (address => UserInfo) public userInfo;
- address public migrator;
- address public vault;
- bool public active;
- bool public initialized;
- uint256 public stakingFee;
- uint256 public priceMultiplier;
- uint256 public lastRewardBlock;
- uint256 public buybackBalance;
- uint256 public buybackRate;
- uint256 public pendingRewards;
- uint256 public rewardsPerShare;
- uint256 public totalShares;

Functions

AnyStakeRegulator has following functions:

- constructor
- initialize
- stabilize
- addReward
- updatePool
- _updatePool
- claim
- _claim
- deposit
- _deposit
- withdraw
- _withdraw
- migrate
- _migrate
- emergencyWithdraw
- isAbovePeg
- pending

- setStakingFee
- setBuybackRate
- setPriceMultiplier
- setMigrator
- setVault
- setActive

AnyStakeVault

Description

AnyStakeVault is a smart contract for storing tokens.

Imports

AnyStakeVault has following imports:

- IUniswapV2Pair.sol
- IAnyStake.sol
- IVaultMigrator.sol
- IAnyStakeRegulator.sol
- IAnyStakeVault.sol
- AnyStakeUtils.sol

Inheritance

AnyStakeVault contract is IAnyStakeVault, AnyStakeUtils.

Usages

AnyStakeVault contract has following usages:

- using SafeMath for uint256;
- using SafeERC20 for IERC20;

Structs

AnyStakeVault contract has no custom structs.

Enums

AnyStakeVault contract has no custom enums.

Events

AnyStakeVault contract has following events:

- event AnyStakeUpdated(address indexed user, address anystake);

- event RegulatorUpdated(address indexed user, address regulator);
- event MigratorUpdated(address indexed user, address migrator);
- event DistributionRateUpdated(address indexed user, uint256 distributionRate);
- event Migrate(address indexed user, address migrator);
- event DeFiatBuyback(address indexed token, uint256 tokenAmount, uint256 buybackAmount);
- event PointsBuyback(address indexed token, uint256 tokenAmount, uint256 buybackAmount);
- event RewardsDistributed(address indexed user, uint256 anystakeAmount, uint256 regulatorAmount);
- event RewardsBonded(address indexed user, uint256 bondedAmount, uint256 bondedLengthBlocks);

Modifiers

AnyStakeVault contract has following modifiers:

- onlyAuthorized

Fields

AnyStakeVault contract has following parameters:

- address public anystake;
- address public regulator;
- address public migrator;
- uint256 public bondedRewards;
- uint256 public bondedRewardsPerBlock;
- uint256 public bondedRewardsBlocksRemaining;
- uint256 public distributionRate;
- uint256 public lastDistributionBlock;
- uint256 public totalBuybackAmount;
- uint256 public totalPointsBuybackAmount;
- uint256 public pendingRewards;
- uint256 public totalRewardsDistributed;

Functions

AnyStakeVault has following functions:

- constructor
- calculateRewards
- distributeRewards
- getTokenPrice
- buyDeFiatWithTokens

- buyPointsWithTokens
- buyTokenWithTokens
- migrate
- addBondedRewards
- setDistributionRate
- setMigrator
- setAnyStake
- setRegulator

AnyStake

Description

AnyStake is a smart contract for tokens staking.

Imports

AnyStake has following imports:

- IDeFiatPoints.sol
- IAnyStake.sol
- IAnyStakeMigrator.sol
- IAnyStakeVault.sol
- AnyStakeUtils.sol

Inheritance

AnyStake contract is IAnyStake, AnyStakeUtils.

Usages

AnyStake contract has following usages:

- using SafeMath for uint256;
- using SafeERC20 for IERC20;

Structs

AnyStake contract has following structs:

- struct UserInfo

uint256 amount;

uint256 rewardDebt;

uint256 lastRewardBlock;

address stakedToken;

address lpToken;

uint256 totalStaked;

uint256 allocPoint;

uint256 rewardsPerShare;

uint256 lastRewardBlock;

uint256 vipAmount;

uint256 stakingFee;

Enums

AnyStake contract has no custom enums.

Events

AnyStake contract has following events:

- event Initialized(address indexed user, address vault);
- event Claim(address indexed user, uint256 indexed pid, uint256 amount);
- event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
- event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event Migrate(address indexed user, uint256 indexed pid, uint256 amount);
- event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);
- event PoolAdded(address indexed user, uint256 indexed pid, address indexed stakedToken, address lpToken, uint256 allocPoints);
- event MigratorUpdated(address indexed user, address migrator);
- event VaultUpdated(address indexed user, address vault);
- event PoolAllocPointsUpdated(address indexed user, uint256 indexed pid, uint256 allocPoints);
- event PoolVipAmountUpdated(address indexed user, uint256 indexed pid, uint256 vipAmount);
- event PoolStakingFeeUpdated(address indexed user, uint256 indexed pid, uint256 stakingFee);
- event PointStipendUpdated(address indexed user, uint256 stipend);

Modifiers

AnyStake contract has following modifiers:

- NoReentrant
- onlyVault
- activated

Fields

AnyStake contract has following parameters:

- address public migrator;
- address public vault;
- bool public initialized;
- PoolInfo[] public poolInfo;
- mapping(uint256 => mapping(address => UserInfo)) public userInfo;
- mapping(address => uint256) public pids;
- uint256 public lastRewardBlock;
- uint256 public pendingRewards;
- uint256 public pointStipend;
- uint256 public totalAllocPoint;
- uint256 public totalBlockDelta;
- uint256 public totalEligiblePools;

Functions

AnyStake has following functions:

- constructor
- initialize
- addReward
- updatePool
- _updatePool
- claim
- _claim
- deposit
- _deposit
- withdraw
- _withdraw
- migrate
- _migrate
- emergencyWithdraw
- getPrice
- pending
- poolLength
- addPoolBatch
- addPool
- _addPool

- setMigrator
- setVault
- setPoolAllocPoints
- setPoolVipAmount
- setPoolChargeFee
- setPointStipend

Audit overview

■■■■ Critical

1. [Fixed in 4e91e27] AnyStakeBooster code is not finalized, not tested and not ready for the audit. It should be finalized before secondary audit.
2. [Fixed in 4e91e27] AddReward is an external function in AnyStakeRegulator. It should be accessible ONLY by admin.
3. [Fixed in 4e91e27] emergencyWithdraw function doesn't charge any fee. Any user can call update, claim and emergencyWithdraw so he won't pay any fee.

■■■ High

4. [Not an issue] It's possible to frontrun buybacks, thus, the transaction may fail or have a bad exchange rate. It should be clearly specified by the dev team what should happen in case of buybacks frontrunning.
5. [Fixed in 4e91e27] If Uniswap migrates to a new version - it will be impossible to change router and factory addresses.
6. [Not an issue] AnyStake uses only 1 source for LP token price which potentially could be manipulated via flashloans.

■■ Medium

7. [Not an issue] Deposit function calls claim function on every deposit, however, it can be called only for withdraw to save gas.
8. [Fixed in 4e91e27] Code is not optimized for gas usage.
9. [Fixed in 4e91e27] If a user doesn't have stake in each pool - claimAll function will always revert.

Low

10. [Fixed in 4e91e27] Solidity version is not locked. It's recommended to lock solidity pragma to a specific stable version.
11. [Fixed in 4e91e27] Wrong comment on line 84 of AnyStake - the base should be 1000.

Lowest / Code style / Best Practice

No best practice issues were found.

Conclusion

Smart contracts within the scope was manually reviewed and analyzed with static analysis tools. For the contract high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found 3 critical, 3 high, 3 medium and 2 low severity issues during the initial audit. All of the issues were resolved for the secondary audit.

Disclaimers

Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.