Security Assessment

Mobius Finance

Aug 24th, 2021

Table of Contents

Summary

This report has been prepared for Mobius Finance to discover issues and vulnerabilities in the source code of the Mobius Finance project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Mobius Finance
Description	Mobius Finance's objective is to create a decentralized multi-asset trading protocol that can support any financial instrument, including off-chain assets such as ETF, commodities, stocks, bonds.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/mobiusfinance/contracts
Commit	f980295a5c93b843f51380c04fa6180a63badd02 f980295a5c93b843f51380c04fa6180a63badd02 3ad4c8d678dc7b7236b53da200f27fdc022e9fb3 d37428cdde8cfb80db0066e6db3c9e69d644561c

Audit Summary

Delivery Date	Aug 24, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

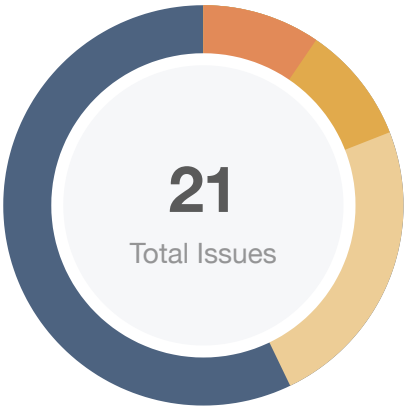
Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	0	2	0
Medium	2	0	0	0	0	2
Minor	5	0	0	0	0	5
Informational	12	0	0	0	0	12
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ESM	base/ExternalStorable.sol	6d9d80101e9ce0de0476c85c9c1ce7340b4991dc8894e4d56934e7fc22c7071b
IMC	base/Importable.sol	4f0b95a4f12b124d0a4222cc7043b7d1498abf97b42f50b7c7c3ad23b5e61b62
CLO	oracles/ChainLinkOracle.sol	be8e7ad5f3fe63d049a6b0a8313f2f57b998ea2c0be47e0b838911dab553893f
MOM	oracles/MobiusOracle.sol	83ee171ab607ca9dcdb133eeec615d1cec3b4cd399932353e05229a5fb37cad5
ASM	storages/AddressStorage.sol	8b1a79fedb7382153b1075a84dbf39025ea99faa11a1d9441a1956d9f909ac8e
ESC	storages/EscrowStorage.sol	d4f0292f6334329c0bfb5566ae628de3158ae1ab97bedc2af522834f18252d96
ESK	storages/ExternalStorage.sol	8401e99dc0ae8a161ab77697a3d93f4ebc93ccccbf662357d81cb49042a03114
ISM	storages/IssuerStorage.sol	a75ace86413cef82b72abdad891751ff23ca0747ebea45dbc0b872e1b8214214
LSM	storages/LiquidatorStorage.sol	8ff148f561fa58cf1a293c89657879975755d94ea8e123b16c3a6cb9cae05751
OSM	storages/OracleStorage.sol	bf48e207b0344b99f54587a967e32837fbf60fc646f1e68f2dab636feb141c12
SSM	storages/SettingStorage.sol	c5e2e85aa01082c7d2262b42311de984bef612c8377afbf6ba08aeb3bec38b94
SSC	storages/StakerStorage.sol	cda2c542a2de09656791681cd5e620a9d55780eefc83df41bc5d860b724a35ef
TSM	storages/TokenStorage.sol	7d1ffbdfb6ea095232fe5bb29e94fed78dfdccca3691ac48da178cfe5d2d925c
TSC	storages/TraderStorage.sol	2831f233c4ecb973388529c864bb2d7bd679b4024eba1c755e64cd8f6ca9969e
APM	AssetPrice.sol	6ff12ebec8342652cdc6d177d2e5a8821bafe11b25bcb8ef7417848f630e432d

ID	File	SHA256 Checksum
DTF	DynamicTradingFee.sol	ee6bbe34227a2d1518a7e7c9b0dbb0bc432117c109051d1b640a12d035318ca6
EMC	Escrow.sol	3cde4898799a3d2da50c86827fd2f827352fc490b86140176ed161437d9e5bf4
IMK	Issuer.sol	b131cea9e4b0968f2e159a690256a93829caea050f9974b1c03fd69694196fc4
LMC	Liquidator.sol	44b987b24b7871a243dc2a525048d1101a44f8d9f6b53483b8524761cd4132c6
MMC	Migrations.sol	4fd6092bdfa8b42f19d535c5ac69c4323b0b894717c699e58d5552eeabd04cd4
MMK	Mobius.sol	63c45f71f5e854b8b14689468987a022b363c1de907a7b69d28750791344b988
MTM	MobiusToken.sol	6aa914f6135201e2523f9dcdede9ab1f638dde5e84fb0b1880611101c4548f29
RMC	Resolver.sol	0e5d1eaca1178dfdfa4c8c80f067d29bb494d2721ecb5420bfd51b2545c628c9
RCM	RewardCollateral.sol	f2b9eb0d764efbbf57529f47073d004c965523e81c70715a0e413db96ffac609
RSM	RewardStaking.sol	99c265fc9397c7c97a25dc9bb4eb206ffa95a240cb445f03db429e6c8dc352b0
RTM	RewardTrading.sol	6579a133ba8f20388dd90b5f51d5c8ee6a5c629bbcea35844a673b011262f27a
SMC	Setting.sol	c2eb4a9e1d4efece2db2ca8833679caf528d62b77d2ee90357bcdf82e2d593fe
SMK	Staker.sol	853feaff8af75470daf6334dce1abf548f4c726ebd705110e85724c5aae118dd
SMP	Synth.sol	d42744befbae41875ab49445f28e2636e99fba0367fa1bca64d129cd0acecbec
TMC	Trader.sol	2e32814ce3f6bdc5248347a6aaa9ba23915b4f1edb275ea99a5cd663cfc228cf

Findings



Critical	**0** (0.00%)
Major	**2** (9.52%)
Medium	**2** (9.52%)
Minor	**5** (23.81%)
Informational	**12** (57.14%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
APM-01	Proper Usage of `public` and `external` Type	Gas Optimization	Informational	Resolved
APM-02	Different Pragma Versions	, Coding Style	Informational	Resolved
DTF-01	Unused Parameter	Coding Style	Informational	Resolved
ESM-01	Missing Zero Address Validation	Logical Issue	Informational	Resolved
IMC-01	Comparison with boolean	Coding Style	Informational	Resolved
LSM-01	Incorrect Check of "addWatch()"	Logical Issue	Minor	Resolved
MOM-01	Price Oracle Setting	Control Flow	Major	Partially Resolved
MTM-01	Total Supply Of $MOT	Logical Issue	Medium	Resolved
MTM-02	Liquidation Settlements	Inconsistency	Medium	Resolved
MTM-03	Potentially excessive permissions	**Centralization / Privilege**	**Major**	Partially Resolved
RCM-01	Missing Emit Events	Gas Optimization	Informational	Resolved
RCM-02	Missing Zero Address Validation	Logical Issue	Informational	Resolved
RCM-03	Potential Reentrancy Issue	Logical Issue	Minor	Resolved
RSM-01	Missing Emit Events	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
RSM-02	Missing Zero Address Validation	Logical Issue	Informational	Resolved
RSM-03	`add()` Function Not Restricted	Volatile Code	Minor	Resolved
RSM-04	Potential Reentrancy Issue	Logical Issue	Minor	Resolved
RTM-01	Missing Emit Events	Gas Optimization	Informational	Resolved
RTM-02	Misleading Naming For Function "swap"	Coding Style	Informational	Resolved
RTM-03	Function "setRouters" Missing Check	Gas Optimization	Informational	Resolved
SMC-01	Costly Operations Inside A Loop	Gas Optimization	Minor	Resolved

APM-01 | Proper Usage of `public` and `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	AssetPrice.sol: 52~55, 67~76	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than `public` functions.

Example: Functions `getPrices` and `getPricesAndStatus` in contract `AssetPrice`.

Recommendation

Consider using the `external` attribute for functions never called from the contract.

Alleviation

The recommendations were applied in commit `8c3b84e8e72d7dc709aef122fe6e4cf4fd5066a3`.

APM-02 | Different Pragma Versions

Category	Severity	Location	Status
, Coding Style	Informational	AssetPrice.sol: 1	Resolved

Description

Contracts use `pragma solidity ^0.8.0;`, `pragma solidity >=0.4.22 <0.9.0;` and `pragma solidity >=0.5.0;` different versions in this protocol. This is not recommended. And pragmas should be locked to specific compiler versions and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, that latest compiler, which may have a higher risker of undiscovered bugs.

```
- >=0.4.22 <0.9.0 (Migrations.sol#2)
- ^0.7.5 (Mobius.sol#2)
- >=0.5.0 (IChainLinkAggregator#2)
```

Recommendation

We recommend avoid using floating and nonuniform pragma versions.

Alleviation

The recommendations were applied in commit `76c04e2710e8daf1fc8410f0f746b588224de7fa`.

DTF-01 | Unused Parameter

Category	Severity	Location	Status
Coding Style	Informational	DynamicTradingFee.sol: 51~54	Resolved

Description

The input parameters `amountInUSD` and `isShort` are unused currently, the function name `getDynamicTradingFeeRate` may be misleading.

Recommendation

Consider using `Setting().getTradingFeeRate(synth)` directly as no dynamic fee currently.

Alleviation

The team has modified and committed in `d6af4b4ea1508dcf5e47e75e43bcb4b22397b0a3`.

ESM-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	base/ExternalStorable.sol: 18~21	Resolved

Description

Functions `constructor` and `setmotProxy` in contract `RewardCollateral.sol`, `setmotProxy` in contract `RewardStaking.sol`, `setStorage` in `ExternalStorable.sol`.

All of them are missing address zero checks.

Recommendation

Consider adding zero address check, for example:

```
function setmotProxy(address _motProxy) external onlyOwner {
    require(_motProxy != address(0), "motProxy is a zero address");
    motProxy = _motProxy;
}
```

Alleviation

The recommendations were applied in commit `f956151f133388a2b4beebbc41d8ee0893db533c`.

IMC-01 | Comparison with boolean

Category	Severity	Location	Status
Coding Style	Informational	base/Importable.sol: 49	Resolved

Description

Performs comparison with a boolean literal false which can be replaced with the negation of the expression to increase the legibility of the codebase.

Recommendation

Consider modifying like below:

```
modifier containAddressOrOwner(bytes32[] memory names) {

    ......

    if (!contain) contain = (msg.sender == owner);
    require(contain, contractName.concat(': caller is not in dependencies'));
    _;
}
```

Alleviation

The recommendations were applied in commit `5c5b2840cd5d1aae85a884a7bc54be8178dcd6ca`.

LSM-01 | Incorrect Check of "addWatch()"

Category	Severity	Location	Status
Logical Issue	Minor	storages/LiquidatorStorage.sol: 15~27	Resolved

Description

Assuming that there is only one account in the current array, when the same account is added again, `_storage[stake][account][0]` represents the index of account in the array, which is equal to 0 at this time, then the same account will be added to the arrays again.

Recommendation

We suggest to modify the judgment condition as `_storage[stake][account][1] > 0`.

Alleviation

The team has modified and committed in `2ddbfbbd9cadad972252c0b097800088e3cbcad3`.

MOM-01 | Price Oracle Setting

Category	Severity	Location	Status
Control Flow	Major	oracles/MobiusOracle.sol: 19~31	Partially Resolved

Description

There are two kinds of oracle models used in the code, which are `ChainLinkOracle` and `MobiusOracle`, moreover, `MobiusOracle` is totally designed by the team. For one asset, the owner has the access to set the corresponding price oracle.

In fact, the asset price provided by the `MobiusOracle` is manually set by the owner. Once the team selected it as the primary price oracle, it implied a huge potential centralization risk.

Recommendation

The `MobiusOracle` should have a strictly audited price model.

Alleviation

[Mobius Finance Team]: We use chainlink mainly, but sometimes we can't get prices on-chain(by chainlink or others), we need an off-chain price-feed model, for example, the price of carbon which we get from charging data provider; Sometimes the price on AMM is too easy to manipulate because of the lack of liquidity. We will not open-source our mobiusOracle code until we find a decentralized way, we will use a proxy contract to visit mobiusOracle to protect us from being attacked.

MTM-01 | Total Supply Of $MOT

Category	Severity	Location	Status
Logical Issue	Medium	MobiusToken.sol	Resolved

Description

In the white paper, the `$MOT` total tokens supply will be one hundred million tokens, no further tokens will be issued, but there is no relevant implementation in the code.

Recommendation

It is recommended to keep the white paper and code implementation consistent.

Alleviation

The recommendations were applied in commit `d81022d08f8fd635881fe8b1376dd60f12a6aaa9`.

MTM-02 | Liquidation Settlements

Category	Severity	Location	Status
Inconsistency	Medium	MobiusToken.sol: 1	Resolved

Description

In the white paper, the collateral will be auctioned by a ten percent discount, four percent of this will be allocated for the reserve auction pool. It is inconsistent with the implementation in the code which will transfer one-third `unstakable` assets to the `LiquidationFeeAddress` address.

Recommendation

It is recommended to keep the white paper and code implementation consistent.

Alleviation

[Mobius Finance Team]: We will set the LiquidationFeeRate to 10%, and then 2/5 belongs to LiquidationFeeAddress, 3/5 belongs to the players.

MTM-03 | Potentially excessive permissions

Category	Severity	Location	Status
Centralization / Privilege	**Major**	MobiusToken.sol: 26~29	Partially Resolved

Description

Function `mint` is merely called by the owner, and it allows the caller to mint tokens to any specified recipient. To improve the trustworthiness of this protocol, any plan to the mint token is better to move to the execution queue of `Timelock` and also add an `emit event`, or make the owner Multi-sig.

```
10  function mint(address account, uint256 amount) external override
containAddress(MINTABLE_CONTRACTS) returns (bool) {
11      _mint(account, amount);
12      return true;
13  }
```

Recommendation

In general, we strongly encourage the centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices.

Indicatively, here are some feasible solutions that would also mitigate the potential risk based on your business flow:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Mobius Team]: Function mint is not called by the owner, it's called by **MINTABLE_CONTRACTS** (RewardCollateral.sol, RewardStaking.sol, RewardTrading.sol), which is a decentralized mechanism.

RCM-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	RewardCollateral.sol: 58~61, 84~91	Resolved

Description

Several sensitive actions are defined without event declarations.

Examples:

`setMOTPerBlock` and `set` in `RewardCollateral.sol`;

`setMOTPerBlock` in `RewardStaking.sol`;

`setMOTPerBlock` in `RewardTrading.sol`.

Recommendation

Consider adding events for sensitive actions, and emit it in the function like below.

```
event _setMOTPerBlock(uint256 _newPerBlock);

function setMOTPerBlock(uint256 _newPerBlock) public onlyOwner {
    massMintPools();
    motPerBlock = _newPerBlock;
    emit _setMOTPerBlock(_newPerBlock);
}
```

Alleviation

The recommendations were applied in commit `c68f59fe090d0c551dea5614b38c57bb32d98caf`.

RCM-02 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	RewardCollateral.sol: 33~43, 49~51	Resolved

Description

Functions `constructor` and `setmotProxy` in contract `RewardCollateral.sol`, `setmotProxy` in contract `RewardStaking.sol`, `setStorage` in `ExternalStorable.sol`.

All of them are missing address zero checks.

Recommendation

Consider adding zero address check, for example:

```
function setmotProxy(address _motProxy) external onlyOwner {
    require(_motProxy != address(0), "motProxy is a zero address");
    motProxy = _motProxy;
}
```

Alleviation

The recommendations were applied in commit `f956151f133388a2b4beebbc41d8ee0893db533c`.

RCM-03 | Potential Reentrancy Issue

Category	Severity	Location	Status
Logical Issue	Minor	RewardCollateral.sol: 158~170	Resolved

Description

There's potential reentrancy issue that the value of `user.amount` and `user.rewardDebt` are updated after the function `safeTransfer()` is called, where `pending` will stay same if there's reentrancy issue caused starting from `safeTransfer(msg.sender, pending);`

Recommendation

We advise the client to adopt `nonReentrant` modifier in openzeppelin library to the function `deposit()` to prevent any reentrancy issue.

Alleviation

The recommendations were applied in commit `cf62b2b96f3d813ac501fcab1ed489eb2c8b4697`.

RSM-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	RewardStaking.sol: 50~53	Resolved

Description

Several sensitive actions are defined without event declarations.

Examples:

`setMOTPerBlock` and `set` in `RewardCollateral.sol`;

`setMOTPerBlock` in `RewardStaking.sol`;

`setMOTPerBlock` in `RewardTrading.sol`.

Recommendation

Consider adding events for sensitive actions, and emit it in the function like below.

```
event _setMOTPerBlock(uint256 _newPerBlock);

function setMOTPerBlock(uint256 _newPerBlock) public onlyOwner {
    massMintPools();
    motPerBlock = _newPerBlock;
    emit _setMOTPerBlock(_newPerBlock);
}
```

Alleviation

The recommendations were applied in commit `c68f59fe090d0c551dea5614b38c57bb32d98caf`.

RSM-02 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	RewardStaking.sol: 32~40, 46~48	Resolved

Description

Functions `constructor` and `setmotProxy` in contract `RewardCollateral.sol`, `setmotProxy` in contract `RewardStaking.sol`, `setStorage` in `ExternalStorable.sol`.

All of them are missing address zero checks.

Recommendation

Consider adding zero address check, for example:

```
function setmotProxy(address _motProxy) external onlyOwner {
    require(_motProxy != address(0), "motProxy is a zero address");
    motProxy = _motProxy;
}
```

Alleviation

The recommendations were applied in commit `f956151f133388a2b4beebbc41d8ee0893db533c`.

RSM-03 | `add()` Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Minor	RewardStaking.sol: 57~72	Resolved

Description

When adding the same LP token more than once. Rewards will be messed up if you do.

The total amount of reward in function `updatePool()` will be incorrectly calculated if the same LP token is added into the pool more than once in function `add()`.

However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation relies on the owner's trust to avoid repeatedly adding the same LP token to the pool, as the owner will only call the function.

Recommendation

We recommend adding the check for ensuring whether the given pool for addition is a duplicate of an existing pool so that the pool addition is only successful when there is no duplicate. This can be done by using a mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice. In addition, consider not using contract `MasterChef` and to use contract `MasterChefV2` instead, since `MasterChefV2` has already solved this issue by adding `nonDuplicated` modifier.

Alleviation

The recommendations were applied in commit `951b0bd3817ec859d337da8e61c00d09303d4fd9`.

RSM-04 | Potential Reentrancy Issue

Category	Severity	Location	Status
Logical Issue	Minor	RewardStaking.sol: 130~146	Resolved

Description

There's potential reentrancy issue that the value of `user.amount` and `user.rewardDebt` are updated after the function `safeTransfer()` is called, where `pending` will stay same if there's reentrancy issue caused starting from `safeTransfer(msg.sender, pending);`

Recommendation

We advise the client to adopt `nonReentrant` modifier in openzeppelin library to the function `deposit()` to prevent any reentrancy issue.

Alleviation

The recommendations were applied in commit `cf62b2b96f3d813ac501fcab1ed489eb2c8b4697`.

RTM-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	RewardTrading.sol: 57~60	Resolved

Description

Several sensitive actions are defined without event declarations.

Examples:

`setMOTPerBlock` and `set` in `RewardCollateral.sol`;

`setMOTPerBlock` in `RewardStaking.sol`;

`setMOTPerBlock` in `RewardTrading.sol`.

Recommendation

Consider adding events for sensitive actions, and emit it in the function like below.

```
event _setMOTPerBlock(uint256 _newPerBlock);

function setMOTPerBlock(uint256 _newPerBlock) public onlyOwner {
    massMintPools();
    motPerBlock = _newPerBlock;
    emit _setMOTPerBlock(_newPerBlock);
}
```

Alleviation

The recommendations were applied in commit `c68f59fe090d0c551dea5614b38c57bb32d98caf`.

RTM-02 | Misleading Naming For Function "swap"

Category	Severity	Location	Status
Coding Style	Informational	RewardTrading.sol: 127~154	Resolved

Description

The function `swap()` does not actually exchange the two assets, but only gives the user a certain amount of trading proceeds, so this naming is misleading.

Recommendation

We propose to change the name to `claim`.

Alleviation

The team has modified the function name to `TradeMing()` in commit `113924dcb1433c153bd189e05d1d996127972ce5`.

RTM-03 | Function "setRouters" Missing Check

Category	Severity	Location	Status
Gas Optimization	Informational	RewardTrading.sol: 52~54	Resolved

Description

There's no sanity check to validate if `_routers` is empty, and the function could be declared `external` to save gas as the input parameter is an array.

Recommendation

we recommend modify like below:

```
function setRouters(address[] memory _routers) external onlyOwner {
    require(_routers.length > 0, "router is empty");
    routers = _routers;
}
```

Alleviation

The recommendations were applied in commit `113924dcb1433c153bd189e05d1d996127972ce5`.

SMC-01 | Costly Operations Inside A Loop

Category	Severity	Location	Status
Gas Optimization	Minor	Stats.sol: 78~84	Resolved

Description

Costly operations inside a loop might waste gas, so optimizations are justified, refer to:
https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Recommendation

Use a local variable to hold the loop computation result. For example in Line58-64 in this contract.

Alleviation

Mobius team heeded this advice, and changed the code in commit `d37428cdde8cfb80db0066e6db3c9e69d644561c`.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.