Security Assessment

ButterSwap III

Aug 17th, 2021

Table of Contents

Summary

This report has been prepared for ButterSwap to discover issues and vulnerabilities in the source code of the ButterSwap III 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	ButterSwap III
Platform	Heco
Language	Solidity
Codebase	https://github.com/butter-swap/butterswap-nft
Commit	3f5f841010aa5ab9e362bfeb7de81aeefed4c22a 98046909586ee42bdb43c00581af8a1c4257a6fa

Audit Summary

Delivery Date	Aug 17, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

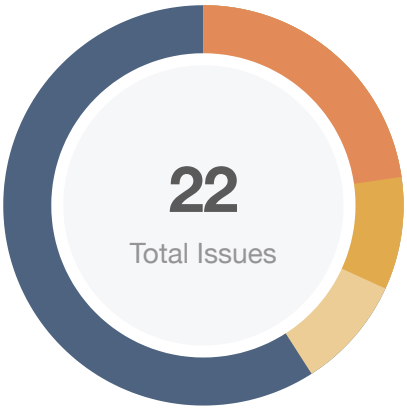
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	5	0	0	5	0	0
Medium	2	0	0	2	0	0
Minor	2	0	0	2	0	0
Informational	13	0	0	1	0	12
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
BBS	BlindBoxStruct.sol	c5540286284d5daa9f988b8bbf63cec3ac1ae1896b58b74b6dcf8376e1a763da
BBT	BlindBoxToken.sol	9ada60ebe9aac50c1c23333320132cab72c6004c23868658bd7a3384ea85a119
BBB	ButterBlindBoxFactory.sol	6ad898e0c72f37b025a195a31c5715ab5c0fc8dea275d1318a28dcebe957efce
BCT	ButterCardToken.sol	ed2cc45f528b4ed205597849e57a17279e7b7225eb782a06cf0ada87d4160831
DTC	DinnerTableChef.sol	5ec29be9d0f5bea29294927e99dbd34623f7af223e558d2384a45ff68f341fb1
RGC	RandomGenerator.sol	dc80fc1fa613f56cf4086b74709a836b515428553fdbae1131f1440b42e8954b

Findings



Critical	**0** (0.00%)
Major	**5** (22.73%)
Medium	**2** (9.09%)
Minor	**2** (9.09%)
Informational	**13** (59.09%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unknown Imported Source File	Volatile Code	Informational	Acknowledged
BBB-01	Privileged ownership in `ButterBlindBoxFactory` contract	**Centralization / Privilege**	**Major**	Acknowledged
BBB-02	Centralized risk in `swapAndSendToFee`	**Centralization / Privilege**	**Medium**	Acknowledged
BBB-03	SafeMath Not Used	Mathematical Operations	Informational	Resolved
BBB-04	3rd party dependencies	Control Flow	Minor	Acknowledged
BBB-05	Redundant comparison against zero	Language Specific	Informational	Resolved
BBB-06	Lack of input validation	Volatile Code	Informational	Resolved
BBB-07	Risk For Weak Randomness	Volatile Code	Minor	Acknowledged
BBS-01	Privileged ownership in `BlindBoxStruct` contract	**Centralization / Privilege**	**Major**	Acknowledged
BBS-02	Variable could be declared as `constant`	Gas Optimization	Informational	Resolved
BBS-03	Declaration Naming Convention	Coding Style	Informational	Resolved
BBS-04	Lack of document for special bonus	Logical Issue	Informational	Resolved

ID	Title	Category	Severity	Status
BBT-01	Limit the Execution of Function `safeMint`	**Logical Issue, Centralization / Privilege**	**Major**	Acknowledged
BBT-02	Misleading Constructor	Volatile Code	Informational	Resolved
BCT-01	Misleading Constructor	Volatile Code	Informational	Resolved
BCT-02	Limit the Execution of Function `safeMint` and `activateCard`	**Logical Issue, Centralization / Privilege**	**Major**	Acknowledged
DTC-01	Centralized risk in `swapAndSendToFee`	**Centralization / Privilege**	**Medium**	Acknowledged
DTC-02	Lack of input validation	Volatile Code	Informational	Resolved
DTC-03	Redundant comparison against zero	Language Specific	Informational	Resolved
DTC-04	Privileged ownership in `DinnerTableChef` contract	**Centralization / Privilege**	**Major**	Acknowledged
RGC-01	Unused variable	Gas Optimization	Informational	Resolved
RGC-02	Make initializer check stricter	Logical Issue	Informational	Resolved

GLOBAL-01 | Unknown Imported Source File

Category	Severity	Location	Status
Volatile Code	Informational	Global	Acknowledged

Description

The imported source files:

1. `../libs/math/SafeMath.sol`
2. `../libs/token/HRC20/IHRC20.sol`
3. `../libs/token/HRC20/SafeHRC20.sol`
4. `../libs/access/Ownable.sol`
5. `../ILuckyLucky.sol`
6. `../libs/token/HRC721/IHRC721Receiver.sol`
7. `../IRandomNumberGenerator.sol`
8. `../libs/utils/Counters.sol`
9. `../libs/token/HRC721/extensions/HRC721Enumerable.sol`

are unknown.

Alleviation

The development team responded as shown below:

1. ../ILuckyLucky.sol and ../IRandomNumberGenerator.sol are same as files in the batch-2 audit.
2. HRC20 HRC721 Ownerable are standard library.

BBB-01 | Privileged ownership in `ButterBlindBoxFactory` contract

Category	Severity	Location	Status
Centralization / Privilege	**Major**	ButterBlindBoxFactory.sol: 19, 65, 70, 75, 80, 85, 90, 95, 99, 107, 116	Acknowledged

Description

The owner of the contract `ButterBlindBoxFactory` has the permission to call:

1. `setAdmin`,
2. `setTreasury`,
3. `setPoolAddress`,
4. `setBurnRate`,
5. `setTreasuryRate`,
6. `setDiscount`
7. `setUseChainLinkRandom`
8. `setMaxCardSlots`
9. `transferCardTokenOwner`
10. `transferBoxTokenOwner`

without obtaining the consensus of the community.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

Alleviation

The development team responded that the owner address will use Multisignature wallets, the admin address will be given to community/board to manage and they will renounce ownership in the futrue.

BBB-02 | Centralized risk in `swapAndSendToFee`

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	ButterBlindBoxFactory.sol: 173	Acknowledged

Description

```
//DinnerTableChef
function unlockSlot() external {
    ...
    butter.safeTransfer(treasury, treasuryFee);
}
```

```
//ButterBlindBoxFactory.sol
function buyBlindBox(
    uint256 boxId,
    uint256 amount
) external {
...
butter.safeTransfer(treasury, treasuryFee);
...
}
```

The `unlockSlot` function of contract `DinnerTableChef` and the function `buyBlindBox` of contract `ButterBlindBoxFactory` call the `butter.safeTransfer` function with the `to` address specified as `treasury`. As a result, over time the `treasury` address will accumulate a significant portion of CAKE tokens. If the `treasury` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

The development team responded as below: the butter transferred to treasury address will be made as a board pool reward for board members every several days or exceeding a certain amount. so the amount won’t be too large.

BBB-03 | SafeMath Not Used

Category	Severity	Location	Status
Mathematical Operations	Informational	ButterBlindBoxFactory.sol: 221~226, 239	Resolved

Description

SafeMath from OpenZeppelin is not used in the following functions which makes them possible for overflow/underflow and will lead to an inaccurate calculation result.

```
uint256 totalPower_ = totalPower(boxId);
uint256 randomNumber = uint256(keccak256(
            abi.encode(
                block.timestamp, block.difficulty,
                priceETH, priceBTC, priceHT,
                block.gaslimit, gasleft(), block.coinbase)
            )) % totalPower_;
```

```
uint256 randomNumber = _randomNumber % totalPower(boxId);
```

The return value of the function `totalPower` can be 0 that will make the calculation failed.

Recommendation

We advise the client to use OpenZeppelin's SafeMath library for all of the mathematical operations.

```
uint256 totalPower_ = totalPower(boxId);
uint256 randomNumber = uint256(keccak256(
            abi.encode(
                block.timestamp, block.difficulty,
                priceETH, priceBTC, priceHT,
                block.gaslimit, gasleft(), block.coinbase)
            )).mod(totalPower_);
```

```
uint256 randomNumber = _randomNumber.mod(totalPower(boxId));
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BBB-04 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	Minor	ButterBlindBoxFactory.sol: 60~62	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `EACAggregatorProxy` protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised that led to assets being lost or stolen.

```
priceFeedBTC = AggregatorV3Interface(0xD5c40f5144848Bd4EF08a9605d860e727b991513);
priceFeedHT = AggregatorV3Interface(0x8EC213E7191488C7873cEC6daC8e97cdbAdb7B35);
priceFeedETH = AggregatorV3Interface(0x5Fa530068e0F5046479c588775c157930EF0Dff0);
```

Recommendation

We understand that the business logic of the `ButterBlindBoxFactory` requires the interaction `EACAggregatorProxy` protocol for acquiring the price of `BTC`, `HT` and `ETH`. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The development team responded that they only use the third party contract to get the prices, and the prices are only used to be as part of seed-parameters to generate a random number.

BBB-05 | Redundant comparison against zero

Category	Severity	Location	Status
Language Specific	Informational	ButterBlindBoxFactory.sol: 81, 86	Resolved

Description

`x >= 0` will be always `true` if `x` is a uint256.

Recommendation

Consider removing redundant comparisons.

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BBB-06 | Lack of input validation

Category	Severity	Location	Status
Volatile Code	Informational	ButterBlindBoxFactory.sol: 56~59	Resolved

Description

The assigned values to address type variables `butter`, `admin`, `blindBox`, and `butterCard` should be verified as non-zero values to prevent error.

Recommendation

Check that the addresses are not zero in the constructor, like below:

```
require(butterCard != address(0));
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BBB-07 | Risk For Weak Randomness

Category	Severity	Location	Status
Volatile Code	Minor	ButterBlindBoxFactory.sol: 203~210	Acknowledged

Description

A self-defined function is used to generate the random number.

Recommendation

Consider mixing a seed value based on the trusted 3rd party random service.

Alleviation

The development team responded that they will use chainlink vrf to generate a random number when chainlink supports heco chain and the `useChainLinkRandom` is for the switching.

BBS-01 | Privileged ownership in `BlindBoxStruct` contract

Category	Severity	Location	Status
Centralization / Privilege	**Major**	BlindBoxStruct.sol: 6, 116, 134, 151, 176, 208	ⓘ Acknowledged

Description

The owner of the contract `BlindBoxStruct` has the permission to call:

1. `pushBox`,
2. `updateBoxPrice`,
3. `pushFamily`,
4. `pushSpecial`,
5. `pushCardMetaData`,

without obtaining the consensus of the community.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

Alleviation

The development team responded that `BlindBoxStruct` is just as the farther contract of `ButterBlindBoxFactory`. And these functions are for the operation using. And some day in the future the admin will be offered to the board.

BBS-02 | Variable could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	BlindBoxStruct.sol: 55, 86~93	Resolved

Description

Variables `pointsDecimal`, `MAX_ENERGY_POINT`, `MAX_RECYCLE_POINT`, `MAX_FAMILY_DINNER_POOL_BONUS`, `MAX_FAMILY_FARMING_BONUS`, `MAX_SPECIAL_DINNER_POOL_BONUS`, `MAX_SPECIAL_FARMING_BONUS` and `MIN_PRICE` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend declaring those variables as `constant`.

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BBS-03 | Declaration Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	BlindBoxStruct.sol: 105, 188	Resolved

Description

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

1. camelCase : Should be applied to function names, argument names, local and state variable names, modifiers
2. UPPER_CASE : Should be applied to constant variables
3. CapWords : Should be applied to contract names, struct names, event names and enums

Recommendation

We advise that the linked event name is adjusted to properly conform to Solidity's naming convention.

```
event NewSpecialPushed(uint256 specialId);
```

```
emit NewSpecialPushed(specialId);
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BBS-04 | Lack of document for special bonus

Category	Severity	Location	Status
Logical Issue	Informational	BlindBoxStruct.sol: 275	Resolved

Description

Lack of documentation in the client's official website(https://docs.butterswap.me/products/nft) for the detailed rules of the special bonus.

Alleviation

The development team heeded our advice and updated the docs.

BBT-01 | Limit the Execution of Function `safeMint`

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	BlindBoxToken.sol: 28	Acknowledged

Description

The owner account can mint nft to anyone at any time by the function `safeMint`. Any compromise to the owner account may allow the hacker to take advantage of this function and eventually damage the contract.

Recommendation

Consider refactoring the code to make the function `safeMint` only be called by the contract `ButterBlindBoxFactory`.

Alleviation

The development team responded that the owner account is ButterBlindBoxFactory, and that would never change unless they need to upgrade the ButterBlindBoxFactory contract. They will make sure the new owner will be the new Factory contract. Users should confirm that the owner's address is the `ButterBlindBoxFactory` contract before using this protocol.

BBT-02 | Misleading Constructor

Category	Severity	Location	Status
Volatile Code	Informational	BlindBoxToken.sol: 25~26	Resolved

Description

The code as below implies that it is a test token:

```
constructor() HRC721("Test Blind Box Token", "TBOX") public {}
```

Recommendation

Considering refactoring the code as below:

```
constructor() HRC721("Butter Blind Box Token", "BBOX") public {}
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BCT-01 | Misleading Constructor

Category	Severity	Location	Status
Volatile Code	Informational	ButterCardToken.sol: 28~29	Resolved

Description

The code as below implies that it is a test token:

```
constructor() HRC721("Test Card Token", "TCARD") public {}
```

Recommendation

Considering refactoring the code as below:

```
constructor() HRC721("Butter Card Token", "BCARD") public {}
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

BCT-02 | Limit the Execution of Function `safeMint` and `activateCard`

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	ButterCardToken.sol: 31~46	Acknowledged

Description

The owner account can mint nft to anyone at any time by the function `safeMint`. The owner account can update activateBlock by the function `activateCard`. Any compromise to the owner account may allow the hacker to take advantage of this function and eventually damage the contract.

Recommendation

Consider refactoring the code to make the function `safeMint` and `activateCard` only be called by the contract `ButterBlindBoxFactory`.

Alleviation

The development team responded that the owner account is ButterBlindBoxFactory, and that would never change unless they need to upgrade the ButterBlindBoxFactory contract. They will make sure the new owner will be the new Factory contract. Users should confirm that the owner's address is the `ButterBlindBoxFactory` contract before using this protocol.

DTC-01 | Centralized risk in `swapAndSendToFee`

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	DinnerTableChef.sol: 316	Acknowledged

Description

```
//DinnerTableChef
function unlockSlot() external {
    ...
    butter.safeTransfer(treasury, treasuryFee);
}
```

```
//ButterBlindBoxFactory.sol
function buyBlindBox(
    uint256 boxId,
    uint256 amount
) external {
...
butter.safeTransfer(treasury, treasuryFee);
...
}
```

The `unlockSlot` function of contract `DinnerTableChef` and the function `buyBlindBox` of contract `ButterBlindBoxFactory` call the `butter.safeTransfer` function with the `to` address specified as `treasury`. As a result, over time the `treasury` address will accumulate a significant portion of CAKE tokens. If the `treasury` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

The development team responded as below: the butter transferred to treasury address will be made as a board pool reward for board members every several days or exceeding a certain amount. so the amount won’t be too large.

DTC-02 | Lack of input validation

Category	Severity	Location	Status
Volatile Code	Informational	DinnerTableChef.sol: 104~110	Resolved

Description

The assigned values to address type variables `butterCard`, `factory`, `butter`, `admin`, `operator`, and `treasury` should be verified as non-zero values to prevent error.

Recommendation

Check that the addresses are not zero in the constructor, like below:

```
require(butterCard != address(0));
```

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

DTC-03 | Redundant comparison against zero

Category	Severity	Location	Status
Language Specific	Informational	DinnerTableChef.sol: 198, 192	Resolved

Description

`x >= 0` will be always `true` if `x` is a uint256.

Recommendation

Consider removing redundant comparisons.

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

DTC-04 | Privileged ownership in `DinnerTableChef` contract

Category	Severity	Location	Status
Centralization / Privilege	**Major**	DinnerTableChef.sol: 384	Acknowledged

Description

The owner of the contract `DinnerTableChef` has the permission to:

1. set `admin` and `treasury`,
2. withdraw the balance of the reward token by calling the function `stopRewardAndEmergencyWithdrawAllButter`,

without obtaining the consensus of the community.

The admin of the contract `DinnerTableChef` has the permission to:

1. set `operator`,`defaultUnlockSlotPrice`, `slotPrice`, `burnRate`, and `treasuryRate`

without obtaining the consensus of the community.

The operator of the contract `DinnerTableChef` has the permission to:

1. update `rewardPerBlock`

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

The development team responded that they'll renounce ownership and give admin to board. The operator permission is combined with the whole ecosystem, everyday 1/15 of total pool butter would be the new reward of this day.

RGC-01 | Unused variable

Category	Severity	Location	Status
Gas Optimization	Informational	RandomGenerator.sol: 69~75	Resolved

Description

Some unused variables are declared. Remove or comment out the variable name.

```
event NewBoxPushed(uint256 boxId, uint256 price, uint256 totalSupply);
event BoxPriceUpdated(uint256 boxId, uint256 price);
event NewFamilyPushed(uint256 familyId);
event newSpecialPushed(uint256 specialId);
event NewCardPushed(uint256 boxId, uint256 level, uint256 totalSupply);
event CardTokenOwnerChanged(address newAddress);
event BoxTokenOwnerChanged(address newAddress);
```

Recommendation

We recommend removing the unused variables in `RandomGenerator.sol`.

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

RGC-02 | Make initializer check stricter

Category	Severity	Location	Status
Logical Issue	Informational	RandomGenerator.sol: 46~47	Resolved

Description

OpenZeppelin has removed `_isConstructor()` check in the `initializer` modifier to make it stricter during construction.

Reference:https://github.com/OpenZeppelin/openzeppelin-contracts/pull/2531/files

Recommendation

Consider removing `_isConstructor()` check in the `initializer` modifier.

Alleviation

The development team heeded our advice and resolved this issue in commit c7c37400efe5c1ada1f3c313af6f31a33cf010e8.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

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.