CERTIK

Preliminary Comments

KeplerSwap

Sept 7th, 2021

Table of Contents

About

Summary

This report has been prepared for KeplerSwap to discover issues and vulnerabilities in the source code of the KeplerSwap 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	KeplerSwap
Platform	BSC
Language	Solidity
Codebase	
Commit	

Audit Summary

Delivery Date	Sept 07, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

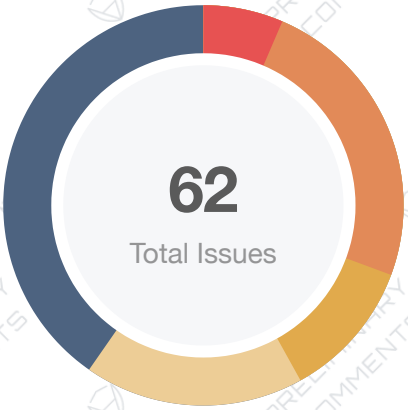
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	4	2	0	2	0	0
Major	15	8	0	3	1	3
Medium	7	2	0	1	0	4
Minor	11	6	0	1	0	4
Informational	25	1	0	2	2	20
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum

Findings



Critical	4 (6.45%)
Major	15 (24.19%)
Medium	7 (11.29%)
Minor	11 (17.74%)
Informational	25 (40.32%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	solc-0.7.2 is not recommended for deployment	Language Specific	Informational	Resolved
CFP-01	Potential Reentrancy Issue	Logical Issue	Major	Pending
CFP-02	Reward update issue	**Centralization / Privilege**	**Major**	Pending
CRC-01	Privileged ownership	**Centralization / Privilege**	**Minor**	Pending
CRC-02	`isContract` Is Not Safe	Volatile Code	Medium	Pending
CRC-03	Missing Input Validation	Volatile Code	Medium	Pending
CRC-04	Check Effect Interaction Pattern Violated	Volatile Code	Minor	Pending
CRC-05	Centralized Control of `voteWiners`	**Centralization / Privilege**	**Major**	Pending
CRC-06	`reward` May Exceed Available Tokens	Volatile Code	Major	Pending
CRY-01	Unused State Variable	Gas Optimization	Informational	Resolved
CRY-02	State Variables That could be Declared Immutable	Gas Optimization	Informational	Resolved
CRY-03	`getPairTokenPrice` May Be Attacked by `flash-loan`	Logical Issue	Major	Acknowledged
CRY-04	Use `revert` Instead of `require`	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
CRY-05	Use `days` Instead Of Calculation	Gas Optimization	Informational	Resolved
CRY-06	Missing Emit Events	Coding Style	Informational	Resolved
CRY-07	Missing Check Function Caller	Control Flow	Minor	Resolved
FDA-01	Privileged Ownership	**Centralization / Privilege**	**Major**	Pending
FDA-02	`owner` Can withdraw Tokens From Contract	**Centralization / Privilege**	**Critical**	Pending
FDP-01	Unnecessary Check	Gas Optimization	Informational	Acknowledged
FDP-02	Functions Should Be Declared External	Gas Optimization	Informational	Resolved
FDP-03	`addXXXDestination` May Add Repeated `destination`	Logical Issue	Minor	Acknowledged
FDP-04	Missing Emit Events	Coding Style	Informational	Acknowledged
FDP-05	Tokens May Be Locked In Contract	Logical Issue	Major	Acknowledged
INV-01	Incorrect Calculation	Volatile Code	Minor	Resolved
INV-02	Missing Invalid `_inviter` Check	Logical Issue	Medium	Resolved
INV-03	Potential Reentrancy Issue	Logical Issue	Major	Pending
KTP-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
KTP-02	Missing Emit Events	Coding Style	Informational	Partially Resolved
KTP-03	Everyone Can Mint KeplerToken	Volatile Code	Major	Resolved
KTP-04	Unused Function Parameter Can Be Removed	Language Specific	Informational	Resolved
KTP-05	Functions Should Be Declared External	Gas Optimization	Informational	Resolved
KTP-06	No Upper Limit Check For currentSnapshotId	Volatile Code	Medium	Resolved
LPA-01	Lack of Input Validation	Volatile Code	Minor	Pending
LPA-02	Mismatch Between Code and Comment	Volatile Code	Minor	Pending

ID	Title	Category	Severity	Status
LPA-03	Privileged Ownership	**Centralization / Privilege**	**Minor**	Pending
LPP-01	Use `days` Instead Of Calculation	Gas Optimization	Informational	Resolved
LPP-02	Unused State Variable	Gas Optimization	Informational	Resolved
LPP-03	`onlyOwnerOrSelf` Is No Need	Gas Optimization, Volatile Code	Informational	Resolved
LPP-04	Functions Should Be Declared External	Gas Optimization	Informational	Resolved
LPP-05	`countUser` Should Be Ended At `countAt` ?	Logical Issue	Informational	Resolved
LPP-06	`bestUser` Is Determined By The Order Of Winner	Logical Issue	Critical	Acknowledged
LPP-07	Non EOA Account May Retry To Win The Game	Logical Issue	Major	Partially Resolved
LPP-08	`addRewardToken` May Add Repeated Tokens	Logical Issue	Major	Resolved
MCA-01	Potential Sandwich Attacks	Volatile Code	Critical	Pending
MCA-02	Deposit Token Doesn't Match Withdraw Token	Inconsistency	Major	Pending
MCA-03	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	Informational	Pending
MCA-04	Potential Reentrancy Issue	Volatile Code	Major	Pending
MCA-05	Privileged Ownership	**Centralization / Privilege**	**Minor**	Pending
MCP-01	Missing Emit Events	Volatile Code	Informational	Partially Resolved
MCP-02	Functions Should Be Declared External	Gas Optimization	Informational	Resolved
MCP-03	Use `days` Directly	Gas Optimization, Coding Style	Informational	Resolved
MCP-04	Use `revert` Instead of `require`	Gas Optimization, Coding Style	Informational	Resolved

ID	Title	Category	Severity	Status
MCP-05	Writing Error(shares)	Logical Issue	Minor	Resolved
MCP-06	Tokens May Be Locked In Contract	Logical Issue	Medium	Acknowledged
MCP-07	Use `RATIO` As Before	Coding Style	Informational	Resolved
MCP-08	Unused Function Parameter Can Be Removed	Language Specific	Informational	Resolved
MCP-09	`_beforeDepositOrWithdraw` Never Be Called	Logical Issue	Medium	Resolved
MCP-10	`_inviter`'s Info Should Be Updated	Logical Issue	Major	Resolved
MCP-11	Caller Can Get Tokens Doesn't Belong To it	Logical Issue	Critical	Acknowledged
MCP-12	No Upper Limit Check For currentSnapshotId	Volatile Code	Medium	Resolved
RAN-01	Weak PRNG	Volatile Code	Major	Acknowledged
USR-01	No Restrict To Repeat Calling `registe`	Logical Issue	Minor	Resolved

GLOBAL-01 | solc-0.7.2 is not recommended for deployment

Category	Severity	Location	Status
Language Specific	Informational	Global	Resolved

Description

solc-0.7.2 is not recommended for deployment.

Recommendation

Deploy with any of the following Solidity versions:

0.5.16 - 0.5.17 0.6.11 - 0.6.12 0.7.5 - 0.7.6 Use a simple pragma version that allows any of these versions.

CFP-01 | Potential Reentrancy Issue

Category	Severity	Location	Status
Logical Issue	Major	farm/CrycleFarm.sol	Pending

Description

The following functions have state updates and event emits after external calls and thus is vulnerable to reentrancy attack.

- doHardWork()
- claim()

Recommendation

We advise using the Checks-Effects-Interactions Pattern to avoid the risk.

CFP-02 | Reward update issue

Category	Severity	Location	Status
Centralization / Privilege	Major	farm/CrycleFarm.sol: 29	Pending

Description

When updating the reward in function `dohardwork`, the reward is overwritten by variable `_amount`. The previous amount is discarded.

Recommendation

We recommend adding the amount to the original one instead of replacing it.

CRC-01 | Privileged ownership

Category	Severity	Location	Status
Centralization / Privilege	Minor	farm/Crycle.sol: 51, 55, 187, 231	Pending

Description

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

1. addPair,
2. removePair,
3. startVote,
4. doCount,

without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. 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, e.g. Multisignature wallets. 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 a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

CRC-02 | `isContract` Is Not Safe

Category	Severity	Location	Status
Volatile Code	Medium	farm/Cycle.sol: 108	Pending

Description

`isContract` returns true if `account` is a contract, but if it returns false, we can not assume that an address is not an contract. Comments before the function `isContract`.

```
1       * It is unsafe to assume that an address for which this function returns
2       * false is an externally-owned account (EOA) and not a contract.
3       *
4       * Among others, `isContract` will return false for the following
5       * types of addresses:
6       *
7       *  - an externally-owned account
8       *  - a contract in construction
9       *  - an address where a contract will be created
10       *  - an address where a contract lived, but was destroyed
```

CRC-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Medium	farm/Cycle.sol: 187	Pending

Description

In the function `startVote`, parameters `beginAt`, `countAt` and `finishAt` are not checked.

Recommendation

We advise the client to add input validation, and it's better to have a fixed voting period.

CRC-04 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Volatile Code	Minor	farm/Cycle.sol: 249	Pending

Description

The order of external call/transfer and storage manipulation should follow the check-effect-interaction pattern.

Recommendation

We advise the client to follow the check-effect-interaction pattern .LINK

CRC-05 | Centralized Control of `voteWiners`

Category	Severity	Location	Status
Centralization / Privilege	Major	farm/Cycle.sol: 231	Pending

Description

The contract owner can call `doCount` to set `voteWiners`, the `userVote` doesn't have relationship with `voteWiners`.

CRC-06 | `reward` May Exceed Available Tokens

Category	Severity	Location	Status
Volatile Code	Major	farm/Crycle.sol: 196	Pending

Description

In function `startVote`, `reward` is initialized by `sds.balanceOf(address(this))`. Although the function checks if the last vote finishes, the reward for last vote may not be claimed.

CRY-01 | Unused State Variable

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/Crycle.sol (ce3c968): 42	Resolved

Description

State variable `user` initialised in contractor, but never used.

Recommendation

We advise to remove unused variable.

CRY-02 | State Variables That could be Declared Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/Crycle.sol (ce3c968): 43~47	Resolved

Description

State variables should be declared immutable to save gas if they are initialized in the constructor and never changed.

```
IMasterChef public masterChef;
IKeplerPair public pair;
IERC20 public busd;
IERC20 public sds;
IKeplerFactory public factory;
```

Recommendation

We recommend changing the codes like the below examples:

```
IMasterChef public immutable masterChef;
IKeplerPair public immutable pair;
IERC20 public immutable busd;
IERC20 public immutable sds;
IKeplerFactory public immutable factory;
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

CRY-03 | `getPairTokenPrice` May Be Attacked by `flash-loan`

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/Crycle.sol (ce3c968): 73	Acknowledged

Description

We find that the `getPairTokenPrice` relies on price calculations that are based on-chain, meaning that they would be susceptible to flash-loan attacks by manipulating the price of given pairs to the attacker's benefit.

Recommendation

If a project requires price references, it needs to be careful of flash loans that might manipulate token prices. To prevent this from happening, we recommend the following suggestions:

1. Use a reliable on-chain price oracle, such as Chainlink.
2. Use Time-Weighted Average Price (TWAP). The TWAP represents the average price of a token over a specified time frame. If an attacker manipulates the price in one block, it will not affect too much on the average price.
3. If the business model allows, restrict the function caller to be a non-contract/EOA address.
4. Flash loans only allow users to borrow money within a single transaction. If the contract use cases are allowed, force critical transactions to span at least two blocks.

Alleviation

Team: prohibited contract calls.

However, we did not found the code to prohibit contract calls.

CRY-04 | Use `revert` Instead of `require`

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/Crycle.sol (ce3c968): 125	Resolved

Description

In function `startVote`, `require` statement can be replaced by `revert`.

Recommendation

we advise to use `revert` directly.

```
1   revert("last vote not finish");
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

CRY-05 | Use `days` Instead Of Calculation

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/Crycle.sol (ce3c968): 133~134	Resolved

Description

Use `days` directly, instead of calculation

Recommendation

We recommend using `days` like below.

```
countAt: block.timestamp + 5 days,
   finishAt: block.timestamp + 7 days
```

Alleviation

The code is removed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

CRY-06 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/farm/Crycle.sol (ce3c968): 101, 113	Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `createCrycle()`
- `addCrycle()`
- `startVote()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

CRY-07 | Missing Check Function Caller

Category	Severity	Location	Status
Control Flow	Minor	contracts/farm/Crycle.sol (ce3c968): 138	Resolved

Description

Function `doVote` can be called by anyone. It seems it's valid called by users only in `userCrycle`.

Recommendation

We recommend adding restriction to this function.

```
1  require(userCrycle[msg.sender] != 0, "not in crycle");
```

FDA-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	farm/FeeDispatcher.sol: 56, 66, 94, 104, 132, 142	Pending

Description

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

1. addDefaultDestination,
2. delDefaultDestination,
3. addTokenDestination,
4. delTokenDestination,
5. addRelateDestination,
6. delRelateDestination,
7. **determin how many tokens been transferred to which destination**

without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. 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, e.g. Multisignature wallets. 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 a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

FDA-02 | `owner` Can withdraw Tokens From Contract

Category	Severity	Location	Status
Centralization / Privilege	Critical	farm/FeeDispatcher.sol: 196	Pending

Description

The contract owner can withdraw tokens from contract. Anyone who compromises the owner account can drain all tokens from the contract.

Recommendation

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. 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, e.g. Multisignature wallets. 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 a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

FDP-01 | Unnecessary Check

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/FeeDispatcher.sol (ce3c968): 47, 85, 123, 61, 99, 136	Acknowledged

Description

In function `checkXXXDestination`, `totalPercent` is `uint`, so `totalPercent >= 0` is always true. In function `delXXXDestination`, `id` is `uint`, so `id >= 0` is always true.

Recommendation

We advise to remove these unnecessary checks.

Alleviation

Team: We still want to add this judgment reserve just in case.

FDP-02 | Functions Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/FeeDispatcher.sol (ce3c968): 149	Resolved

Description

Functions that are never called internally within the contract should have external visibility. For example, `doHardWork`.

Recommendation

We recommend changing the visibility of the aforementioned functions to external.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

FDP-03 | `addXXXDestination` May Add Repeated `destination`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/farm/FeeDispatcher.sol (ce3c968): 50, 88, 126	Acknowledged

Description

The function `addXXXDestination` allows the owner to add the same `destination` many times.

Recommendation

We advise to check repeated `destination` in `addXXXDestination` function.

Alleviation

Team: The same address will have a different percentage of the transaction fee income.

FDP-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/farm/FeeDispatcher.sol (ce3c968): 51, 70, 89, 108, 127, 146	Acknowledged

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `addDefaultDestination()`
- `delDefaultDestination()`
- `addTokenDestination()`
- `delTokenDestination()`
- `addRelateDestination()`
- `delRelateDestination()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

Team: Since we don't count this information, we don't consider adding events for now.

FDP-05 | Tokens May Be Locked In Contract

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/FeeDispatcher.sol (ce3c968): 150	Acknowledged

Description

In function `doHardWork`, `amount` tokens are transferred to the contract, but fewer (`totalPercent`/`TOTAL_PERCENT`<1) tokens are transferred out, so some tokens may be locked in the contract.

Recommendation

We recommend the client utilize these tokens correctly.

Alleviation

Team: Added the function to withdraw tokens manually. Since the contract will only keep tokens that cannot be transferred because the amount is too small, these tokens cannot be processed properly, to prevent locking in the contract. We manually transfer the tokens to the owner.

INV-01 | Incorrect Calculation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/farm/Inviter.sol (ce3c968): 23	Resolved

Description

In function `doHardWork`, `profits[_inviter][_token]` calculation is incorrect.

```
1   profits[_inviter][_token] = profits[_user][_token].add(_amount);
```

Recommendation

We advise to correct the calculation.

```
1   profits[_inviter][_token] = profits[_inviter][_token].add(_amount);
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

INV-02 | Missing Invalid `_inviter` Check

Category	Severity	Location	Status
Logical Issue	Medium	contracts/farm/Inviter.sol (ce3c968): 22	Resolved

Description

In function `doHardWork`, `_inviter` may not exist, so tokens will be transferred to `address(0)`.

Recommendation

We recommend adding sanity check for `_inviter`.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

INV-03 | Potential Reentrancy Issue

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/Inviter.sol (ce3c968): 26	Pending

Description

There's potential reentrancy issue that the value of `profits[msg.sender][_token]` is updated after the function `safeTransfer()` is called, where `profits[msg.sender][_token` will stay same if there's reentrancy issue caused starting from `safeTransfer(msg.sender, profits[msg.sender][_token);`

Recommendation

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

Alleviation

ReentrancyGuard contract is added, but `nonreentrant` modifier is not used.

KTP-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	contracts/token/KeplerToken.sol (ce3c968): 3	Resolved

Description

The contract has an unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for `^0.6.12`, the version can be locked at `0.6.12`.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

KTP-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/token/KeplerToken.sol (ce3c968): 23, 31	Partially Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setSnapshotCreateCaller()`
- `createSnapshot()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function like below:

```
    event SetSnapshotCreateCaller(address indexed Caller, address indexed newCaller);

    function setSnapshotCreateCaller(address _snapshotCreateCaller) external onlyOwner {
        address createCaller = snapshotCreateCaller;
        snapshotCreateCaller = _snapshotCreateCaller;
        emit SetSnapshotCreateCaller(createCaller, snapshotCreateCaller);
    }
```

Alleviation

`NewSnapshot` added in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

KTP-03 | Everyone Can Mint KeplerToken

Category	Severity	Location	Status
Volatile Code	Major	contracts/token/KeplerToken.sol (ce3c968): 28	Resolved

Description

Function `mint` is public to anyone, without access limit, so anyone can call it to mint tokens.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`, `onlyOwner` can mint tokens now.

KTP-04 | Unused Function Parameter Can Be Removed

Category	Severity	Location	Status
Language Specific	Informational	contracts/token/KeplerToken.sol (ce3c968): 47	Resolved

Description

In function `_beforeTokenTransfer`, `amount` parameter is not used. Remove or comment out the unused variable name to silence a Compiler warning.

Recommendation

We advise that the `amount` can be removed or commented out.

```
1     function _beforeTokenTransfer(address from, address to, uint256 /* amount */)
internal override {
2         //if (false) {
3         //    amount;
4         //}
5         if (currentSnapshotId == 0) {
6             return;
7         }
8         if (userSnapshotId[from] < currentSnapshotId) {
9             userSnapshotAmount[from] = balanceOf(from);
10             userSnapshotId[from] = currentSnapshotId;
11         }
12         if (userSnapshotId[to] < currentSnapshotId) {
13             userSnapshotAmount[to] = balanceOf(to);
14             userSnapshotId[to] = currentSnapshotId;
15         }
16     }
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

KTP-05 | Functions Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/token/KeplerToken.sol (ce3c968): 27	Resolved

Description

Functions that are never called internally within the contract should have external visibility. For example, `mint`.

Recommendation

We recommend changing the visibility of the aforementioned functions to external.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

KTP-06 | No Upper Limit Check For currentSnapshotId

Category	Severity	Location	Status
Volatile Code	Medium	contracts/token/KeplerToken.sol (ce3c968): 31	Resolved

Description

In function `createSnapshot`, there is no upper limit check for `id`, if `id` is set to MAX_UINT256, `createSnapshot` can not be called anymore.

Recommendation

We advise to increase the `id` one by one.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

LPA-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	farm/LuckyPool.sol: 80~129	Pending

Description

If the length of input array `users` is smaller than maximum value in the mapping `perPairNum`, the function would revert.

Recommendation

We recommend checking the length of array `users` in function `openLuckyPool`.

LPA-02 | Mismatch Between Code and Comment

Category	Severity	Location	Status
Volatile Code	Minor	farm/LuckyPool.sol	Pending

Description

The `OPEN_WAIT` and `CLAIM_WAIT` is 5 minutes and 1 hour inside the contract, while the comment claim to be 1 hour and 3 days respectively.

Recommendation

We recommend change the comment to match the code.

LPA-03 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Minor	farm/LuckyPool.sol: 159, 80, 49	Pending

Description

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

1. setRandom,
2. beginLuckyPool,
3. openLuckyPool,

without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. 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, e.g. Multisignature wallets. 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 a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

LPP-01 | Use `days` Instead Of Calculation

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/LuckyPool.sol (ce3c968): 89~90, 96	Resolved

Description

Use `days` Instead Of Calculation

Recommendation

We recommend using `days` like below.

```
poolInfo[_currentLuckyId].countAt = block.timestamp + 2 days;
poolInfo[_currentLuckyId].finishAt = block.timestamp + 5 days;
```

Alleviation

The issue is fixed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

LPP-02 | Unused State Variable

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/LuckyPool.sol (ce3c968): 20	Resolved

Description

State variable `user` initialised in contractor, but never used.

Recommendation

We advise to remove unused variable.

Alleviation

The variable is removed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

LPP-03 | `onlyOwnerOrSelf` Is No Need

Category	Severity	Location	Status
Gas Optimization, Volatile Code	Informational	contracts/farm/LuckyPool.sol (ce3c968): 47	Resolved

Description

Modifier `onlyOwnerOrSelf` checks `msg.sender` should be `owner` or `address(this)`, there is no external function call from `address(this)`, so it's no need to check `address(this)`.

Recommendation

We advise to use `onlyOwner` instead of `onlyOwnerOrSelf`.

Alleviation

The modifier is removed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

LPP-04 | Functions Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/LuckyPool.sol (ce3c968): 52, 56, 84, 99	Resolved

Description

Functions that are never called internally within the contract should have external visibility. For example, `addRewardToken`, `nextLuckyId`, `delRewardToken`, openLuckyPool `and` countUser`.

Recommendation

We recommend changing the visibility of the aforementioned functions to external.

Alleviation

The issue is resolved in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

LPP-05 | `countUser` Should Be Ended At `countAt`?

Category	Severity	Location	Status
Logical Issue	Informational	contracts/farm/LuckyPool.sol (ce3c968): 102	Resolved

Description

In function `countUser`, end time is at `_poolInfo.finishAt`, is it designed?

```
1  require(_poolInfo.openAt != 0 && block.timestamp >= _poolInfo.openAt &&
block.timestamp <= _poolInfo.finishAt, "not the right time");
```

Alleviation

The function is removed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

LPP-06 | `bestUser` Is Determined By The Order Of Winner

Category	Severity	Location	Status
Logical Issue	Critical	contracts/farm/LuckyPool.sol (ce3c968): 153	Acknowledged

Description

In function `claim`, winers can claim there rewards, the `bestUser` is the thrid winner; The order of winner can be affected by the order of transactions included in a block, and the order of transactions can be manipulated by the miner, so miner will win the `bestUser`.

Recommendation

We advise to not use the order to determine `bestUser`.

Alleviation

Team: We do not consider nodes behaving maliciously for now.

LPP-07 | Non EOA Account May Retry To Win The Game

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/LuckyPool.sol (ce3c968): 138	Partially Resolved

Description

Eve may use the contract to calls the `claim` function, and check if it is the winner, if false, it will revert the transaction, and retry it later until it wins the game.

Recommendation

We advise to allow only EOA to call `claim` function.

```
1  require(tx.origin == msg.sender, "non EOA");
```

Alleviation

The client fixed this issue by using function `isContract` in openzeppelin libraries. However, `isContract` will return false for the following types of addresses:

- an externally-owned account
- a contract in construction
- an address where a contract will be created
- an address where a contract lived, but was destroyed

LPP-08 | `addRewardToken` May Add Repeated Tokens

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/LuckyPool.sol (ce3c968): 53	Resolved

Description

The function `addRewardToken` may allow the owner to add the same token many times; it will cause incorrect balances in `poolRewardAmount`.

Recommendation

We advise to check repeated tokens in `addRewardToken` function.

Alleviation

Function `addRewardToken` has been removed in commit 7a8cc099bae6e9b29e0dace8f711e731ff31f1b9.

MCA-01 | Potential Sandwich Attacks

Category	Severity	Location	Status
Volatile Code	Critical	farm/MasterChef.sol: 292, 298, 304	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding/removing liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The `withdraw` function calls the `_pair.burn()` without requiring minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the withdraw amount is large:

- `_pari.burn()`

Recommendation

We recommend setting reasonable minimum output amounts when calling `withdraw` function.

MCA-02 | Deposit Token Doesn't Match Withdraw Token

Category	Severity	Location	Status
Inconsistency	Major	farm/MasterChef.sol: 230	Pending

Description

Users deposit lptokens but withdraw underlying tokens of the lptoken; the lptokens are burned for swapping underlying tokens.

Recommendation

We recommend keeping deposited tokens and withdrawn tokens consistent.

MCA-03 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	Informational	farm/MasterChef.sol: 290	Pending

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

MCA-04 | Potential Reentrancy Issue

Category	Severity	Location	Status
Volatile Code	Major	farm/MasterChef.sol: 271, 386, 412	Pending

Description

The following functions have state updates and event emits after external calls and thus is vulnerable to reentrancy attack.

- withdraw()
- claimMine()
- claimInviteMine()

Recommendation

We advise client to consider to adopt following `nonReentrant` modifier on above mentioned functions:

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol

MCA-05 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Minor	farm/MasterChef.sol: 70	Pending

Description

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

1. setSnapshotCreateCaller,

without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the owner account's private key to avoid any potential risks of being hacked. 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, e.g. Multisignature wallets. 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 a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

MCP-01 | Missing Emit Events

Category	Severity	Location	Status
Volatile Code	Informational	contracts/farm/MasterChef.sol (ce3c968): 63, 283	Partially Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setSnapshotCreateCaller()`
- `createSnapshot()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function like below:

```
    event SetSnapshotCreateCaller(address indexed Caller, address indexed newCaller);

    function setSnapshotCreateCaller(address _snapshotCreateCaller) external
onlyOwner {
        address createCaller = snapshotCreateCaller;
        snapshotCreateCaller = _snapshotCreateCaller;
        emit SetSnapshotCreateCaller(createCaller, snapshotCreateCaller);
    }
```

Alleviation

createSnapshot event added

MCP-02 | Functions Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/farm/MasterChef.sol (ce3c968): 80, 98	Resolved

Description

Functions that are never called internally within the contract should have external visibility. For example, `doMiner` and `doInviteMiner`.

Recommendation

We recommend changing the visibility of the aforementioned functions to external.

MCP-03 | Use `days` Directly

Category	Severity	Location	Status
Gas Optimization, Coding Style	Informational	contracts/farm/MasterChef.sol (ce3c968): 174, 176, 178	Resolved

Description

Solidity supports the unit of `days`, use it directly.

Recommendation

We recommend using `days` directly.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-04 | Use `revert` Instead of `require`

Category	Severity	Location	Status
Gas Optimization, Coding Style	Informational	contracts/farm/MasterChef.sol (ce3c968): 180, 374, 403	Resolved

Description

In function `getType`, `require` statement can be replaced by `revert`.

Recommendation

we advise to use `revert` directly.

```
1   revert("illegal lockType");
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-05 | Writing Error(shares)

Category	Severity	Location	Status
Logical Issue	Minor	contracts/farm/MasterChef.sol (ce3c968): 194, 241	Resolved

Description

In function `inviteDeposit`, there is a writing error, `_userInfo.amount` should be `_userInfo.shares`.

```
1   inviterUserInfo[_pair][_inviter].shares = _userInfo.amount.add(_shares);
```

Recommendation

Correct the error like this below:

```
1   inviterUserInfo[_pair][_inviter].shares = _userInfo.shares.add(_shares);
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-06 | Tokens May Be Locked In Contract

Category	Severity	Location	Status
Logical Issue	Medium	contracts/farm/MasterChef.sol (ce3c968): 88, 106	Acknowledged

Description

In function `doMiner`, `doInviteMiner`, tokens are transferred to the contract; if `_poolInfo.totalShares == 0` is true, these tokens can not be distributed to pool users, and the caller doesn't even know this issue.

Recommendation

We recommend the client solve this issue.

Alleviation

Team: From a business point of view, if total shares is 0 and there are no user locks, then there should be no liquidity and there will be no trades. Also, if total shares==0, then there is no user locking positions, which would not make liquidity classified.

MCP-07 | Use `RATIO` As Before

Category	Severity	Location	Status
Coding Style	Informational	contracts/farm/MasterChef.sol (ce3c968): 362~363, 370~371, 391~392, 399~400	Resolved

Description

`RATIO` is a constant defined in the contract; it's a good choice to use `RATIO` instead of `1e18`.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-08 | Unused Function Parameter Can Be Removed

Category	Severity	Location	Status
Language Specific	Informational	contracts/farm/MasterChef.sol (ce3c968): 298	Resolved

Description

In function `_beforeDepositOrWithdraw`, `amount` parameter is not used. Remove or comment out the unused variable name to silence a Compiler warning.

Recommendation

We advise that the `amount` can be removed or commented out.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-09 | `_beforeDepositOrWithdraw` Never Be Called

Category	Severity	Location	Status
Logical Issue	Medium	contracts/farm/MasterChef.sol (ce3c968): 298	Resolved

Description

The function `_beforeDepositOrWithdraw` is an internal function and never be called in the contract. Function `_beforeTokenTransfer`, in KeplerToken contract, is a hook function called by underlaying `ERC20._transfer`.

Recommendation

We advise to redesign and fix this issue.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-10 | `_inviter`'s Info Should Be Updated

Category	Severity	Location	Status
Logical Issue	Major	contracts/farm/MasterChef.sol (ce3c968): 189	Resolved

Description

In function `inviteDeposit`, `_inviter`'s information should be updated not `_user`'s.

```
1   inviterClear(_pair, _user, _poolInfo, _userInfo);
```

Recommendation

We advise to check the logic and fix this issue.

```
1   inviterClear(_pair, _inviter, _poolInfo, _userInfo);
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

MCP-11 | Caller Can Get Tokens Doesn't Belong To it

Category	Severity	Location	Status
Logical Issue	Critical	contracts/farm/MasterChef.sol (ce3c968): 217	Acknowledged

Description

In function `deposit`, `inviterClear(_pair, msg.sender)` will increase `inviterUserInfo[_pair][_user].tokenxPending`; if `inviterPoolInfo[_pair].tokenxAccPerShare` increased between last update time and the current time calling `deposit` function.

Alleviation

Team: Inviter is getting the same mining rewards as regular users, so update on pending is required.

MCP-12 | No Upper Limit Check For currentSnapshotId

Category	Severity	Location	Status
Volatile Code	Medium	contracts/farm/MasterChef.sol (ce3c968): 283	Resolved

Description

In function `createSnapshot`, there is no upper limit check for `id`, if `id` is set to MAX_UINT256, `createSnapshot` can not be called anymore.

Recommendation

We advise to increase the `id` one by one.

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

RAN-01 | Weak PRNG

Category	Severity	Location	Status
Volatile Code	Major	farm/Random.sol	Acknowledged

Description

Weak PRNG due to a modulo on `block.timestamp, block.difficulty, msg.sender, block.coinbase` and block.number`. These can be influenced by miners to some extent, so they should be avoided.

Recommendation

Do not use `block.timestamp, block.difficulty, msg.sender, block.coinbase` and block.number` as a source of randomness

USR-01 | No Restrict To Repeat Calling `registe`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/user/User.sol (ce3c968): 17	Resolved

Description

Users can repeat calling `registe`, which may cause data inconsistence.

Recommendation

We advise to add `require` statement as below.

```
1  require(inviter[msg.sender] == address(0), "registed");
```

Alleviation

Fixed in `c3bb09ead56b54d14e679dbd3edb48090fb3f0c4`.

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.

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.

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.