CERTIK

Security Assessment

Open Leverage Protocol

Jun 24th, 2021

Table of Contents

Summary

This report has been prepared for Open Leverage Protocol smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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.

Addtionally, this audit is based on a premise that all external contracts were implemented safely.

The security assessment resulted in 26 findings that ranged from major 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Open Leverage Protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/OpenLeverageDev/openleverage-contracts
Commit	<da19c97f472fe4e3ed99279e62c1949f788c7fe0>

Audit Summary

Delivery Date	Jun 24, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Total Issues	26
● Critical	0
● Major	2
● Medium	0
● Minor	5
● Informational	19
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
ALC	contracts/Adminable.sol	b9b9eca1da3ec75db94672a0b90ffcb784075cad238c8f03205a9e944cab5136
CDL	contracts/ControllerDelegator.sol	fafa3bdb9c82e2b2fbdf08d17da46b234b0a587cf7d3eda50a64a4ffc064d183
CIL	contracts/ControllerInterface.sol	ab374638f09a9a55cf309ba1b21b948baa56e4ac8d3adb3eef5a5cac0bba4f9c
CVL	contracts/ControllerV1.sol	255b7445ce016dc2011d93e43ed0919e4d57976a7af0dbe5dbb99bce6ecf3aa9
DIL	contracts/DelegateInterface.sol	0a7d219017e2abecb6d8de95fd3dd58c9d3e6167f77b3fbc25b76d5f7229f6dd
DIC	contracts/DelegatorInterface.sol	c1d6b5c6eef55b030e7132cb1d7648c876f6de2274e790fd1bd68a4be42479d6
DCL	contracts/DexCaller.sol	d0c1c3666cd20a20fcf3f7a8c6cabe5c3420a5e8d7beef0381de1c2e4eedb1f5
MLC	contracts/Migrations.sol	8a6b38936c738a0e612391ee231f39352cc8878f4a5b41c05f0895fb662b3fd6
OLE	contracts/OLETokenLock.sol	48741e21065148b7d80aec944951bec7c225c1c7af7cc2953a321ae44811d55a
OLD	contracts/OpenLevDelegator.sol	da30eebced696e4b34e10ba99b2e95cd5886e603aa52d17ffd6307f996210792
OLI	contracts/OpenLevInterface.sol	a3527bea717351be8309762789ec484144a0f4c1cf9a886f461048c290cbd240
OLV	contracts/OpenLevV1.sol	ebaaa2e82a22e097bd49d4fa92806e46d9ee5c88ead1f27583f43f68911f8c70
RLC	contracts/Referral.sol	f1d3f772fbc001a9d3b4d601818e0c9f4c8b79929cdcd1fb701786a78b938598
RDL	contracts/ReferralDelegator.sol	e664fb4c768686d529008e3227e3f04dfeaa5efc0cfb3d8b7f741bd2b25403de
RIL	contracts/ReferralInterface.sol	d4aedb423a4e35cfc44940f0be675e6ea1c471802f3d0d213d5e996d7d161d24
RLK	contracts/Reserve.sol	f87c64968497ec42d3bfeed68b5b13c4cc3e340db393af2878b74df9f7492336
TLC	contracts/Treasury.sol	96457c4d898745c8c249f1b6a08f079bdadb33adf1e83c605b6e65e092efdeef
TDL	contracts/TreasuryDelegator.sol	3491caba96df9449d0044cc34cd0c7be5b6d2328c992895219cf56ffe217b0da
TIL	contracts/TreasuryInterface.sol	9339f69d2d4a783be4a4f57a4aac0965b28ce92104911d92642597c828686798
TLK	contracts/Types.sol	ef12e3b841f24e76b7b7b362404b71345dea7b501e42991dc919d74c0b5331b1

ID	file	SHA256 Checksum
IUV	contracts/dex/IUniswapV2Callee.sol	286f6d7bced5f9f59c98d4ce614313a3c746c7c4091d6b5dc22da2b3317b2f26
IUF	contracts/dex/IUniswapV2Factory.sol	0a38fb30aa55451d2d6c974c1d97ffac1f9fa0d07eca8d3d5df35e0eafe055b7
IUP	contracts/dex/IUniswapV2Pair.sol	9ea77ebc7fba9b238d93315148036dc0f5006af6a105c9a50dfd48280ba0e06c
POI	contracts/dex/PriceOracleInterface.sol	93964765861b2a5afdf0aae920964a45ab3b6271651bafd309ccfa8b893f3b00
POV	contracts/dex/PriceOracleV2.sol	b5bf761b22610a86304c661394163ea3038016ce3b6773d4daad7b2f01a12f12
FPL	contracts/farming/FarmingPool.sol	215e6d08deb6a362930dc69e073a34d9e7827adf4aaac57c69fc94c65714bc53
GAL	contracts/gov/GovernorAlpha.sol	25ca5949ce37297c62a09efbea33271954838db9e53bc4e7a5f2af0ebddf7483
OLT	contracts/gov/OLEToken.sol	e8de286db912483c8199920ab5fa840fc585aab5ed5d8970a5e4a506f9964895
TLP	contracts/gov/Timelock.sol	d0b1ab2e95cde234f7845371e037aaf87e7075415b9397e71b48528bc40ebbc6
IRM	contracts/liquidity/InterestRateModel.sol	645a0bc98352af26f5cb341d62e0194e906ad5b9e03cbe9a506df2a2b5730e6c
JRM	contracts/liquidity/JumpRateModel.sol	148a8172a49584e8055eba5c264f906a549449abe5acdd4b90a90688b971aa0e
LPL	contracts/liquidity/LPool.sol	6a5d1f475e42329f96207037670eb55af78db71223e56dffa3fcb16c75a1bfe0
LPD	contracts/liquidity/LPoolDelegator.sol	4bee0e86d1c2381445e018cf884068d1ad3936e93721f401f6cc91696469bb71
LPI	contracts/liquidity/LPoolInterface.sol	cc979339fc855db10852e292c0f3caef6d86c3030a6a1544069ce9e692d635ef

Privileged Functions

The project contains the following privileged functions that are restricted by several central roles. They are used to modify the contract configurations and address attributes. We grouped these functions below:

Owner:

[FarmingPool.sol]

- setRewardsDistribution(): set the user to `RewardsDistribution` role.

Guardian

[GovernorAlpha.sol]

- __abdicate(): set 0 address to `guardian` role.
- __acceptAdmin: accept the `admin` role.

RewardsDistribution

[FarmingPool.sol]

- notifyRewardAmount(): notify the reward amount or extend the activity cycle.

Admin

[OLEToken.sol]

- mint(): mint token to `account` and move delegates.

[Adminable.sol]

- setPendingAdmin(): set the user to be `pendingAdmin`.

[LPool.sol]

- initialize: initialize all state variables.
- setController: set `controller` address.
- setBorrowCapFactorMantissa: set `borrowCapFactorMantissa` address.
- setInterestParams: set interest params.

[Referral.sol]

- setRate(): set the date of referral reward.
- setOpenLev: set the address of `openLev`.

- initialize(): initialize `openLev` variable.

[Reserve.sol]

- transfer(): transfer `oleToken` to `to`.

[Treasury.sol]

- setDevFundRatio(): set the ratio of `devFund`.
- setDev(): set `dev` address.

[OpenLevV1.sol]

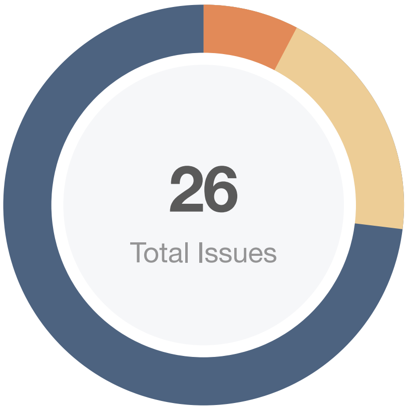
- initialize(): initialize all state variables.
- setDefaultMarginRatio(): set the default ratio of margin.
- setMarketMarginLimit(): set the ratio of margin in market.
- setFeesRate(): set the ratio of transaction fees.
- setInsuranceRatio(): set the ratio of insurance.
- setController(): set the address of `controller`.
- setPriceOracle(): set the address of `priceOracle`.
- setUniswapFactory(): set the address of `uniswapFactory`.
- setReferral(): set the address of `referral`.
- moveInsurance(): withdraw insurance.

Dev

[Treasury.sol]

- devWithdraw(): withdraw `devFund`.

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**2** (7.69%)
🟨	**Medium**	**0** (0.00%)
🟨	**Minor**	**5** (19.23%)
🟦	**Informational**	**19** (73.08%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
ALC-01	Modify The Calling Permission Of The `acceptAdmin` Function	Logical Issue	● Minor	✓ Resolved
CVL-01	Boolean Equality	Coding Style	● Informational	⊗ Declined
CVL-02	Integer Overflow Risk	Mathematical Operations	● Informational	Partially Resolved
FPL-01	Lack Of Verification Of Account Balance	Logical Issue	● Minor	✓ Resolved
GAL-01	Proper Usage of `public` And `external` Type	Gas Optimization	● Informational	✓ Resolved
GAL-02	Boolean Equality	Coding Style	● Informational	✓ Resolved
GAL-03	Ignored Return Values In GovernorAlpha Functions	Logical Issue	● Informational	✓ Resolved
LPL-01	Incorrect Function Return Value	Logical Issue	● Major	✓ Resolved
LPL-02	Integer Overflow Risk	Mathematical Operations	● Informational	⊗ Declined
LPL-03	Missing Zero Address Validation	Logical Issue	● Informational	✓ Resolved
OLE-01	Check Effect Interaction Pattern Violated	Logical Issue	● Informational	✓ Resolved
OLE-02	Lack Of Verification Of Parameters	Logical Issue	● Informational	✓ Resolved
OLE-03	Missing Zero Address Validation	Logical Issue	● Informational	✓ Resolved
OLV-01	Incorrect Source Of `liquidateVars` Variable	Logical Issue	● Major	✓ Resolved

ID	Title	Category	Severity	Status
OLV-02	Missing Emit Events	Coding Style	Informational	Resolved
OLV-03	Meaningless Parameter	Coding Style	Informational	Resolved
OLV-04	Inaccurate Referral Fee Calculation	Logical Issue	Minor	Resolved
POV-01	Way To Update Price	Logical Issue	Minor	Declined
RLC-01	Boolean Equality	Coding Style	Informational	Resolved
RLC-02	Integer Overflow Risk	Mathematical Operations	Informational	Resolved
RLC-03	Lack Of Verification Of Account Balance	Gas Optimization	Informational	Resolved
RLC-04	Local Variable Shadowing	Logical Issue	Informational	Resolved
RLC-05	Lack Of Verification For Referral Rate	Logical Issue	Informational	Resolved
RLK-01	Lack Of Verification Of `amount`	Logical Issue	Minor	Resolved
RLK-02	Missing Zero Address Validation	Logical Issue	Informational	Resolved
TLC-01	Missing Zero Address Validation	Logical Issue	Informational	Resolved

ALC-01 | Modify The Calling Permission Of The `acceptAdmin` Function

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/Adminable.sol: 28	✓ Resolved

Description

According to the logic of the contract, if `admin` calls the `acceptAdmin` function before calling the `setPendingAdmin` function incorrectly, it will cause address(0) to become `admin`. And we think it should be that `pendingAdmin` calls the `acceptAdmin` function to accept `admin`.

Recommendation

Consider modifying the function like below:

```
function acceptAdmin() external virtual {
        require(msg.sender == pendingAdmin,"only pendingAdmin can accept admin");
        address oldAdmin = admin;
        address oldPendingAdmin = pendingAdmin;
        admin = pendingAdmin;
        pendingAdmin = address(0);
        emit NewAdmin(oldAdmin, admin);
        emit NewPendingAdmin(oldPendingAdmin, pendingAdmin);
    }
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

CVL-01 | Boolean Equality

Category	Severity	Location	Status
Coding Style	● Informational	contracts/ControllerV1.sol: 78	⊗ Declined

Description

`marketLiqDistribution[marketId]` is a `bool` type variable, which can be directly used as an expression result in require.

Recommendation

Consider modifying the condition like below:

```
    if (!marketLiqDistribution[marketId]) {return;}
```

Alleviation

No alleviation.

CVL-02 | Integer Overflow Risk

Category	Severity	Location	Status
Mathematical Operations	● Informational	contracts/ControllerV1.sol: 100, 112, 184, 201, 226, 110, 156, 159, 162, 215, 306, 104	Partially Resolved

Description

Using `+`,`-`,`/` in the method directly to calculate the value of the variable may overflow. `SafeMath` provides functions to verify overflow, and it is safer to use the functions provided.

Recommendation

Using the functions in `SafeMath` library for mathematical operations.

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

FPL-01 | Lack Of Verification Of Account Balance

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/farming/FarmingPool.sol: 158	✓ Resolved

Description

When the `reward` is more than the balance of the contract, it will cause incorrect `rewardRate` to be calculated and users fail to withdraw reward. We recommend adding verification of account balance.

Recommendation

Consider add verification of account balance to ensure that the reward is sufficient. For example:

```
function notifyRewardAmount(uint256 reward)
    external override onlyRewardDistribution updateReward(address(0)){
       if(){...}
       else(){...}
       uint balance = oleToken.balanceOf(address(this));
       require(rewardRate<=balance.div(duration)),'balance is not enough');
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

GAL-01 | Proper Usage of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	● Informational	contracts/gov/GovernorAlpha.sol: 140, 180, 197, 207, 254, 258, 290, 295	✓ Resolved

Description

`public` functions that are never called by the contract could be declared `external`.

Recommendation

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

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

GAL-02 | Boolean Equality

Category	Severity	Location	Status
Coding Style	● Informational	contracts/gov/GovernorAlpha.sol: 274	✓ Resolved

Description

`receipt.hasVoted` is a `bool` type variable, which can be directly used as an expression result in require.

Recommendation

Consider modifying the require like below:

```
    require(!receipt.hasVoted, "GovernorAlpha::_castVote: voter already voted");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

GAL-03 | Ignored Return Values In GovernorAlpha Functions

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/gov/GovernorAlpha.sol: 180, 197	✓ Resolved

Description

The GovernorAlpha contract allows successful proposals to be queued and executed through the Timelock contract. The corresponding functions in Timelock return data that may be useful for users, but the data is ignored by the GovernorAlpha contract and is not returned to end users. This may make it more difficult for users to debug and understand whether calls were successful. The relevant functions in the GovernorAlpha contract are: • execute • queue

Recommendation

Add return statements to `execute`, `queue` functions, so that the return values from the Timelock contract can be exposed to users instead of being silently dropped.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

LPL-01 | Incorrect Function Return Value

Category	Severity	Location	Status
Logical Issue	● Major	contracts/liquidity/LPool.sol: 552	✓ Resolved

Description

The function calls other internal functions, when these internal functions are executed incorrectly, the function returns the error code as borrowIndex, which will affect the entire project.

Recommendation

Consider using `require` to verify the result of these internal functions. For example:

```
    require (mathErr == MathError.NO_ERROR,"calc fail");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

LPL-02 | Integer Overflow Risk

Category	Severity	Location	Status
Mathematical Operations	● Informational	contracts/liquidity/LPool.sol: 269, 900	ⓧ Declined

Description

Using `-` in the method directly to calculate the value of the variable may overflow. `SafeMath` provides a function to verify overflow, and it is safer to use the function provided.

Recommendation

Using the `sub` function in `SafeMath` library for mathematical operations.

Alleviation

No alleviation.

LPL-03 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/liquidity/LPool.sol: 33	✓ Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

initialize():

```
require(underlying_ != address(0), "underlying_ address cannot be 0");
require(controller_ != address(0), "controller_ address cannot be 0");
```

transferTokens():

```
require(dst != address(0), "dst address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLE-01 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/OLETokenLock.sol: 33	✓ Resolved

Description

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

Recommendation

We advice client to check if storage manipulation is before the external call/transfer operation by considering following modification:

```
function release(address beneficiary) external {
    uint256 currentTransfer = transferableAmount(beneficiary);
    uint256 amount = token.balanceOf(address(this));
    require(amount > 0, "no amount available");
    require(amount >= currentTransfer, "transfer out limit exceeds ");
    releaseVars[beneficiary].released =
                releaseVars[beneficiary].released.add(currentTransfer);
     token.transfer(beneficiary, currentTransfer);
}
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLE-02 | Lack Of Verification Of Parameters

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/OLETokenLock.sol: 21	✓ Resolved

Description

Lack of verification of the length of `startTimes` and `endTimes`, add the verification to ensure that all array parameters have the same length.

Recommendation

Consider adding the verification of the length of `startTimes` and `endTimes`. For example:

```
require(beneficiaries.length == amounts.length
        && beneficiaries.length == startTimes.length
        && beneficiaries.length == endTimes.length,
        "array length must be same");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLE-03 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/OLETokenLock.sol: 44	✓ Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

```
require(beneficiary != address(0), "beneficiary address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLV-01 | Incorrect Source Of `liquidateVars` Variable

Category	Severity	Location	Status
Logical Issue	● Major	contracts/OpenLevV1.sol: 336	✓ Resolved

Description

`liquidateVars` is a local variable modified with `memory`, which has not been initialized or assigned, so the value of `liquidateVars.borrowed` is always 0, which affects the judgment of the condition.

Recommendation

Consider initializing or assigning `liquidateVars` variable to ensure the correctness of the attribute value.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLV-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	contracts/OpenLevV1.sol: 408, 412, 416, 420, 424	✓ Resolved

Description

Some functions should be able to emit event as notifications to customers because they change the status of sensitive variables.This suggestion is not limited to these codes, but also applies to other similar codes.

Recommendation

Consider adding an emit after changing the status of variables.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLV-03 | Meaningless Parameter

Category	Severity	Location	Status
Coding Style	● Informational	contracts/OpenLevV1.sol: 262	✓ Resolved

Description

The passed-in value of parameter `closeAmount` is always 0, `marketValueCurrent` represents the market value of the assets currently held by the user, `pnl` represents the value of the user's assets loss or gain compared with the value when the position was opened. In other words, `closeAmount` should not be subtracted from `marketValueCurrent`.

Recommendation

Consider removing the `closeAmount` parameter.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

OLV-04 | Inaccurate Referral Fee Calculation

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/OpenLevV1.sol: 362	✓ Resolved

Description

The platform charges fees based on the size of the transaction. The `fees` include insurance and referral fees. Insurance should be deducted from the cost when calculating the referral fee.

Recommendation

We recommend deducting insurance when calculating the referral fee. For example:

```
(referralReward, refereeDiscount) = referral.calReferralReward(msg.sender, referrer, fees.sub(newInsurance), token);
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

POV-01 | Way To Update Price

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/dex/PriceOracleV2.sol	ⓧ Declined

Description

Flash loans are a way to borrow large amounts of money for a certain fee. The requirement is that the loans need to be returned within the same transaction in a block. If not, the transaction will be reverted.

An attacker can use the borrowed money as the initial funds for an exploit to enlarge the profit and/or manipulate the token price in the decentralized exchanges.

We find that the `PriceOracleV2` contract rely on price calculations that are based on fixed numerical algorithm, 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 methods:

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.

Alleviation

No alleviation.

RLC-01 | Boolean Equality

Category	Severity	Location	Status
Coding Style	● Informational	contracts/Referral.sol: 23, 52	✓ Resolved

Description

`isActive` is a `bool` type variable, which can be directly used as an expression result in require.

Recommendation

Consider modifying like below:

registerReferrer:

```
require(!account.isActive, "Already registered");
```

calReferralReward:

```
if (referrerAcct.isActive) { ... }
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLC-02 | Integer Overflow Risk

Category	Severity	Location	Status
Mathematical Operations	● Informational	contracts/Referral.sol: 76, 82~83	✓ Resolved

Description

Using `+` in the method directly to calculate the value of the variable may overflow. `SafeMath` provides a function to verify overflow, and it is safer to use the function provided.

Recommendation

Using the `add()` function in `SafeMath` library for mathematical operations.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLC-03 | Lack Of Verification Of Account Balance

Category	Severity	Location	Status
Gas Optimization	● Informational	contracts/Referral.sol: 68	✓ Resolved

Description

When the user's account balance is 0, the transfer operation is meaningless. We recommend adding verification of the account balance.

Recommendation

Consider adding verification of the account balance. For example:

```
require(withdrawAmt>0,"balance is 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLC-04 | Local Variable Shadowing

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/Referral.sol: 77	✓ Resolved

Description

Local variable `refereeDiscount` shadows `ReferralStorage.refereeDiscount`.

Recommendation

Consider renaming the local variable `refereeDiscount` that shadow another component.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLC-05 | Lack Of Verification For Referral Rate

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/Referral.sol: 111	✓ Resolved

Description

Adding the verification to ensure that sum of all referral rates do not exceed 100%.

Recommendation

We recommend adding the verification as below:

```
    require(_firstLevelRate.add(_secondLevelRate).add(_refereeDiscount) <= 100,
"Invalid params");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLK-01 | Lack Of Verification Of `amount`

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/Reserve.sol: 38	✓ Resolved

Description

When the `admin` incorrectly calls the `transfer` function (amount=0) before the beginning of the vesting, `lastupdate` will be updated to `block.timestamp`, which will cause `lastupdate` to be earlier than `vestingBegin`, the vesting will start early, and the actual vesting amount will be greater than `vestingAmount` finally.

Recommendation

Consider adding restrictions to ensure the `amount` greater than 0. For example:

```
    require(amount > 0,"amount is 0!");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

RLK-02 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/Reserve.sol: 19, 38	✓ Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

constructor():

```
require(_admin != address(0), "_admin address cannot be 0");
require(_oleToken != address(0), "_oleToken address cannot be 0");
```

transfer():

```
require(to != address(0), "to address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

TLC-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/Treasury.sol: 19	✓ Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

```
require(_oleToken != address(0), "_oleToken address cannot be 0");
require(_sharingToken != address(0), "_sharingToken address cannot be 0");
require(_dev != address(0), "_dev address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit bc583d35ed29b866d1b0f84942224ed3730ad9f3.

Appendix

Finding Categories

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.

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 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 without CertiK's prior written consent.

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.

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.

CERTIK