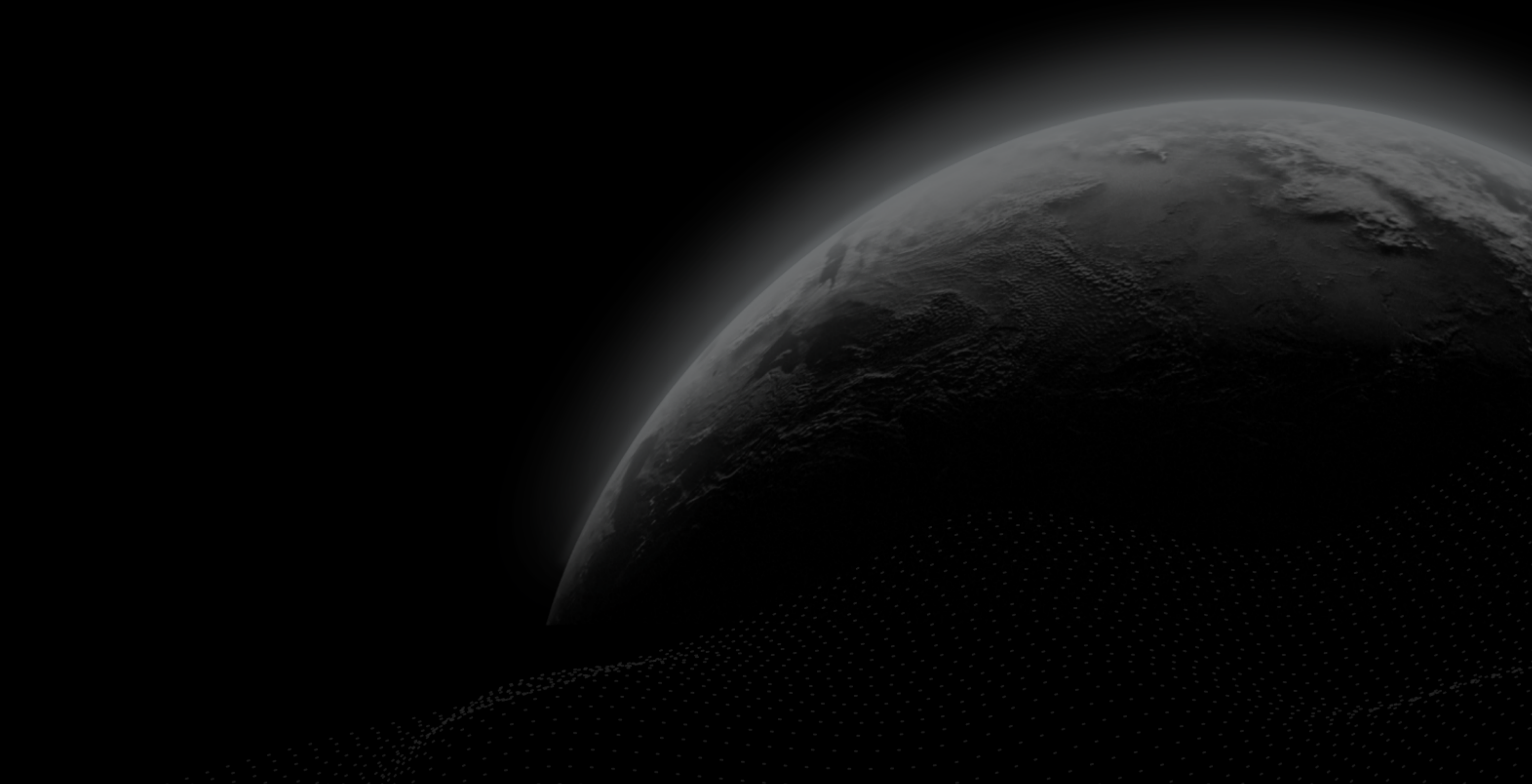
CERTIK

Security Assessment

LightDAO II

CertiK Verified on Apr 20th, 2023



CertiK Verified on Apr 20th, 2023

LightDAO II

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEX	Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 04/20/2023	N/A

CODEBASE

https://github.com/Light-Ecosystem/light-dao/tree/8bb73950b3bb2b85daac2d203640ea5cdfd3811a

https://github.com/Light-Ecosystem/swap-

...View All

COMMITS

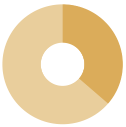
8bb73950b3bb2b85daac2d203640ea5cdfd3811a

eb70734264b46194eb3f7f11335a01588298c3cc

5709f10575c24eedaa1ce813b67936dd63c7274b

...View All

Vulnerability Summary



11	8	0	1	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥 0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧 0	Major		Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨 4	Medium	2 Resolved, 1 Partially Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 7	Minor	6 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦 0	Informational		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | LIGHTDAO II

CODEBASE | LIGHTDAO II

Repository

https://github.com/Light-Ecosystem/light-dao/tree/8bb73950b3bb2b85daac2d203640ea5cdfd3811a https://github.com/Light-Ecosystem/swap-core/tree/5709f10575c24eedaa1ce813b67936dd63c7274b https://github.com/Light-Ecosystem/swap-periphery/tree/4f3c9f29a2c88973f884dac6bf5cafd806a67ee3
https://github.com/Light-Ecosystem/light-vest-escrow/tree/ca1be4ccc153839f391eed6480e5b2579cb62967

Commit

8bb73950b3bb2b85daac2d203640ea5cdfd3811a

eb70734264b46194eb3f7f11335a01588298c3cc

5709f10575c24eedaa1ce813b67936dd63c7274b

4f3c9f29a2c88973f884dac6bf5cafd806a67ee3

ca1be4ccc153839f391eed6480e5b2579cb62967

AUDIT SCOPE | LIGHTDAO II

15 files audited ● 2 files with Acknowledged findings ● 2 files with Partially Resolved findings ● 4 files with Resolved findings ● 7 files without findings

ID	File	SHA256 Checksum
● URL	UniswapV2Router01.sol	162e48778447da691b64455565c33841a00f21383ee54c4686fb22cb914b862c
● URE	UniswapV2Router02.sol	6c57e5073d5f6c9fe90106d97f4557b7689beed6a839a210eba10cb0a4c72a42
● LSB	feeDistributor/LightSwapBurner.sol	dee1c1ba2f8febdf155c2e67889157b2a7dd67b30f614a44b000a41ada0ce07c
● UBD	feeDistributor/UnderlyingBurner.sol	8cc8dd39cd75c2e33213d35c1d3482badc038775cff6dabba6083a5b9ff0a488
● FDD	feeDistributor/FeeDistributor.sol	e1e871782ba098e26cf0d79724629901003d17ac65056d4cf7318cdbdb5d9e5a
● GFD	feeDistributor/GombocFeeDistributor.sol	9af241304e810c0205eec08bcb5f086601be77b3152accab7b6dc74e51c85ad3
● UFL	UniswapV2Factory.sol	3c43a1593640d49ed640534de76155bdf81f242df033a949917e43183063e3ae
● UPL	UniswapV2Pair.sol	cd2304ceefbf34b064ae6db1a68bfb8b18eb61776a3d75fec983e756007325a1
● BMD	feeDistributor/BurnerManager.sol	789c9a98690a7bf99fdef710b9522cfc4e4757e21808c7e519b85aa2b83f1bc4
● SFT	feeDistributor/SwapFeeToVault.sol	cd3390cc460bcd420911f7aa0ca8180c46eb0ffe21450816be904ce7f0a927fe
● ATL	ApprovedTokenManager.sol	bfa38bd16d13e91ff65334062f3e652bc46e58b0c7f1a3bad7271dfce0259d41
● CON	Context.sol	988450c340c4332d49161ad31d38be4b7fd319d9b4fefd086db605ca5990ce6f
● OWN	Ownable.sol	571b9374db230f9be0f226295ac99ff667252691c6b8136f5ff88607bf230744
● UVC	UniswapV2ERC20.sol	02d0d3f4ab7e68311b9300b60e45182fb487c2734924567aba9ed7c6d2bda267

ID	File	SHA256 Checksum
● UML	UniswapV2Migrator.sol	b1bf0327cec8556c602174e20f067782c71c051ddb6f45f00459a062402ca374

APPROACH & METHODS | LIGHTDAO II

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

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

REVIEW NOTES | LIGHTDAO II

HOPE Ecosystem is a staking protocol where users can stake their bought HOPE tokens for LT tokens and other rewards.

System Overview

In this protocol, 50% of admin fees charged from light exchange contracts are distributed to VeLT holders via the FeeDistributor contract and distributed to VeLT holders that vote for the Gauge via the GombocFeeDistributor contract.

Fees are distributed weekly. The proportional amount of fees that each user is to receive is calculated based on their veLT balance relative to the total veLT supply or based on their vote gomboc to the total gomboc weight. This amount is calculated at the start of the week. The actual distribution occurs at the end of the week based on the fees that were collected. As such, a user that creates a new vote-lock should expect to receive their first fee payout at the end of the following epoch week.

The available HOPE balance to distribute is tracked via the “token checkpoint”. This is updated at a minimum every 24 hours. Fees that are received between the last checkpoint of the previous week and the first checkpoint of the new week will be split evenly between the weeks.

Notes

The DEX part of the protocol is forked from the Uniswap v2 protocol. For this part, only the differences between the listed commits were reviewed in this audit.

v2-core:

https://github.com/Uniswap/v2-core/tree/ee547b17853e71ed4e0101ccfd52e70d5acded58

https://github.com/Light-Ecosystem/swap-core/tree/5709f10575c24eedaa1ce813b67936dd63c7274b

v2-periphery:

https://github.com/Uniswap/v2-periphery/tree/0335e8f7e1bd1e8d8329fd300aea2ef2f36dd19f

https://github.com/Light-Ecosystem/swap-periphery/tree/4f3c9f29a2c88973f884dac6bf5cafd806a67ee3

The audit scope includes all the other non-forked files and the delta part of the forked files.

DECENTRALIZATION EFFORTS | LIGHTDAO II

Description

In the contract **BurnerManager**, the role **Owner** has authority over the following functions:

- function setBurner(), to set burner of `token` to `burner` address.
- function setManyBurner(), to set burner of `token` to `burner` address.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **FeeDistributor**, the role **Owner** has authority over the following functions:

- function toggleAllowCheckpointToken(), to toggle permission for checkpointing by any account.
- function recoverBalance(), to recover ERC20 tokens from this contract, send tokens in the contract to the emergency address.
- function setEmergencyReturn(), to set the token emergency return address.
- function pause(), to trigger the stopped state.
- function unpause(), to return to normal state.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **GombocFeeDistributor**, the role **Owner** has authority over the following functions:

- function toggleAllowCheckpointToken(), to toggle permission for checkpointing by any account.
- function recoverBalance(), to recover ERC20 tokens from this contract, send tokens in the contract to the emergency address.
- function setEmergencyReturn(), to set the token emergency return address.
- function pause(), to trigger the stopped state.
- function unpause(), to return to normal state.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **LightSwapBurner**, the role **Owner** has authority over the following functions:

- function setRouters(), to set routers.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **SwapFeeToVault**, the role **Owner** has authority over the following functions:

- function pause(), to trigger the stopped state.
- function unpause(), to return to normal state.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **UnderlyingBurner**, the role **Owner** has authority over the following functions:

- function recoverBalance(), to recover ERC20 tokens from this contract, send tokens in this contract to the emergency address.
- function setRouters(), to set routers.
- function setEmergencyReturn(), to set the token emergency return address.
- function pause(), to trigger the stopped state.
- function unpause(), to return to normal state.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **ApprovedTokenManager**, the role **Owner** has authority over the following functions:

- function approveToken(), to approve the token, only approved tokens are allowed to create Uniswap pairs.

Any compromise to the **Owner** account may allow a hacker to take advantage of this authority.

In the contract **UniswapV2Factory**, the role **feeToSetter** has authority over the following functions:

- function setFeeTo(), to set the address to charge the fee.
- function setFeeToSetter(), to set the `feeToSetter` address.
- function setApprovedTokenManager(), to set the `approvedTokenManager` address.
- function setFeeRateNumerator(), to set fee rate numerator.
- function setLightRewardParams(), to set light reward parameters.
- function setPairGomboc(), to set the pair gomboc.

Any compromise to the **feeToSetter** account may allow a hacker to take advantage of this authority.

`UnderlyingBurner` is an upgradeable contract, the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he can change the implementation of the contract and drain tokens from the contract.

Recommendations

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We recommend carefully managing the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

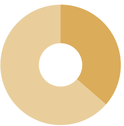
Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Status/Alleviations

FINDINGS | LIGHTDAO II



11	0	0	4	7	0
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for LightDAO II. Through this audit, we have uncovered 11 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
FED-01	Potential Flashloan Attack	Logical Issue	Medium	● Partially Resolved
FED-03	Potential Sandwich Attack	Logical Issue	Medium	● Resolved
GFD-01	`pointsWeight` Is Not Updated	Logical Issue	Medium	● Resolved
UVL-01	Incorrect Import Statement	Logical Issue	Medium	● Acknowledged
FDD-01	`veSupply[_timestamp]` Is Not Updated	Logical Issue	Minor	● Resolved
FDD-02	Incorrect Function Call To `claim` In Function `claimableToken`	Logical Issue	Minor	● Resolved
FED-02	Return Value Not Handled	Logical Issue	Minor	● Resolved
GFD-02	Incorrect Function Call To `claim` In Function `claimableTokens`	Logical Issue	Minor	● Resolved
LEB-01	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
LEU-01	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	● Acknowledged
UPL-01	Lack Of Reasonable Boundary	Volatile Code	Minor	● Resolved

FED-01 | POTENTIAL FLASHLOAN ATTACK

Category	Severity	Location	Status
Logical Issue	● Medium	feeDistributor/LightSwapBurner.sol (light-dao): 58; feeDistributor/UnderlyingBurner.sol (light-dao): 123	● Partially Resolved

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 `burn` rely 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 cautious of flash loans that might manipulate token prices. To minimize the chance of happening, we recommend the client consider following according to the project's business model.

1. Use multiple reliable on-chain price oracle sources, such as Chainlink and Band protocol.
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 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

`[CertiK]` : The team heeded the advice and restricted the function caller to EOA address to partially resolve the finding in the commit 3ee39cb671645acdd78abd0854a6e0bb62313d6a.

FED-03 | POTENTIAL SANDWICH ATTACK

Category	Severity	Location	Status
Logical Issue	● Medium	feeDistributor/LightSwapBurner.sol (light-dao): 71; feeDistributor/UnderlyingBurner.sol (light-dao): 136	● Resolved

Description

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

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `bestRouter.swapExactTokensForTokens()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit a65825aa8e0efc8cf2869476700b768b480c5060.

GFD-01 | `pointsWeight` IS NOT UPDATED

Category	Severity	Location	Status
Logical Issue	● Medium	feeDistributor/GombocFeeDistributor.sol (light-dao): 172, 284, 313, 352	● Resolved

Description

If the point weight of the gomboc at the timestamp is not correctly filled or updated, the functions that calculate the claimable fees may return an incorrect or unexpected result. This could result in users querying the incorrect amount of claimable fees and potentially claiming the wrong amount of tokens.

Recommendation

We recommend calling the function `GombocController.checkpointGomboc` in these functions to ensure using the correct `gomboc` data.

```
function checkpointGomboc(address addr) external override {
    _getWeight(addr);
    _getTotal();
}
```

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit a5de16972eb716f1a035e92923154f17a6102ea9.

UVL-01 | INCORRECT IMPORT STATEMENT

Category	Severity	Location	Status
Logical Issue	● Medium	libraries/UniswapV2LiquidityMathLibrary.sol (swap-periphery): 3 ~4	● Acknowledged

Description

The project builds failed due to incorrect import statements. The contract `IUniswapV2Pair` should be imported from the local Uniswap V2-core lib since the new functions were added.

```
      uint32 feeRateNumerator = IUniswapV2Pair(UniswapV2Library.pairFor(factory,
tokenA, tokenB)).getFeeRateNumerator();
```

```
      IUniswapV2Pair pair = IUniswapV2Pair(UniswapV2Library.pairFor(factory,
tokenA, tokenB));
```

```
      IUniswapV2Pair pair = IUniswapV2Pair(UniswapV2Library.pairFor(factory,
tokenA, tokenB));
```

Recommendation

We recommend reviewing all the import statements in the protocol and fixing the incorrect imports.

Alleviation

`[Light DAO]`: Issue acknowledged. We won't make any changes for the current version. To minimize least change, v2-core will be published and linked as local package.

FDD-01 | `veSupply[_timestamp]` IS NOT UPDATED

Category	Severity	Location	Status
Logical Issue	● Minor	feeDistributor/FeeDistributor.sol (light-dao): 172	● Resolved

Description

The function `vePrecentageForAt` uses the `veSupply[_timestamp]` to calculate the VeLT voting percentage for the user in the gomboc at `_timestamp` . However, the `veSupply[_timestamp] is not updated, which may not get an accurate VeLT voting percentage.

```
169     function vePrecentageForAt(address _user, uint256 _timestamp) external view returns (uint256) {
170         _timestamp = LibTime.timesRoundedByWeek(_timestamp);
171         uint256 veForAtValue = this.veForAt(_user, _timestamp);
172         uint256 supply = veSupply[_timestamp];
173         if (supply == 0) {
174             return 0;
175         }
176         return (veForAtValue * 1e18) / supply;
177     }
```

Recommendation

We recommend reviewing the logic again and calling the function `_checkpointTotalSupply` to update the supply checkpoint.

```
    if (block.timestamp >= timeCursor) {
        _checkpointTotalSupply();
    }
```

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit 677ea0eab04945c4067ac00edb4eb16930e2fbee.

FDD-02 INCORRECT FUNCTION CALL TO `claim` IN FUNCTION `claimableToken`

Category	Severity	Location	Status
Logical Issue	● Minor	feeDistributor/FeeDistributor.sol (light-dao): 375	● Resolved

Description

Based on the function name, we assume this function is used to query the claimable tokens of the user. So the internal function `_claim()` should be called instead of the external function `claim()` .

```
375    function claimableToken(address _addr) external returns (uint256) {
376        return this.claim(_addr);
377    }
```

Recommendation

We recommend reviewing the logic again and ensuring it is as intended, and we also recommend updating the supply checkpoint to ensure the function `claimableToken` returns correct claimable fees.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit 046c6557f93ddac0b0c5a1b458ef9c4cd5a3fa7d.

FED-02 | RETURN VALUE NOT HANDLED

Category	Severity	Location	Status
Logical Issue	● Minor	feeDistributor/FeeDistributor.sol (light-dao): 476; feeDistributor/GombocFeeDistributor.sol (light-dao): 479	● Resolved

Description

The return values of the function `staking` are not properly handled.

```
    function stakingHOPEAndTransfer2User(address to, uint256 amount) internal {
        require(IERC20Upgradeable(token).approve(stHOPE, amount), "APPROVE_FAILED");
        IStakingHOPE(stHOPE).staking(amount, 0, 0, "");
        TransferHelper.doTransferOut(stHOPE, to, amount);
    }
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above to handle both success and failure cases if needed by the business logic.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the finding in the commit d95fd8abae8275dd939627afff291f30adcd195b.

GFD-02 INCORRECT FUNCTION CALL TO `claim` IN FUNCTION `claimableTokens`

Category	Severity	Location	Status
Logical Issue	● Minor	feeDistributor/GombocFeeDistributor.sol (light-dao): 340	● Resolved

Description

Based on the function name, we assume this function is used to query the claimable tokens of the user. So the internal function `_claim()` should be called instead of the external function `claim()` .

```
339      function claimableTokens(address gomboc, address _addr) external whenNotPaused returns (uint256) {
340          return this.claim(gomboc, _addr);
341      }
```

Recommendation

We recommend reviewing the logic again and ensuring it is as intended, and we recommend updating the gomboc checkpoint to ensure the function `claimableToken` returns accurate claimable fees.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit 2c92658dd5a5721a0bee98239191752da12d42aa.

LEB-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	feeDistributor/LightSwapBurner.sol (light-dao): 28; feeDistributor/UnderlyingBurner.sol (light-dao): 144; UniswapV2Factory.sol (swap-core): 82~84; UniswapV2Router01.sol (swap-periphery): 21, 22; UniswapV2Router02.sol (swap-periphery): 24, 25	● Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commits e72303c2ee192ba7414d20aa90e25a814f1ab5e5, 32977d02ed4df30edf3165e2bde828fe8822b9a4 and 915c8c81fa6d24dcb8e580ef30873ed4177829b2.

LEU-01 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	UniswapV2Router01.sol (swap-periphery): 109; UniswapV2Router02.sol (swap-periphery): 113	● Acknowledged

Description

The return value of the transfer()/transferFrom() call is not checked.

```
109         IUniswapV2Pair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity to pair
```

```
113         IUniswapV2Pair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity to pair
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

`[Light DAO]`: Issue acknowledged. We won't make any changes for the current version.

UPL-01 | LACK OF REASONABLE BOUNDARY

Category	Severity	Location	Status
Volatile Code	● Minor	UniswapV2Pair.sol (swap-core): 95	● Resolved

Description

The variable `_feeRateNumerator` does not have reasonable boundaries, so they can be given arbitrary values after deploying.

```
93      function setFeeRateNumerator(uint32 _feeRateNumerator) external {
94          require(msg.sender == factory, 'HopeSwap: FORBIDDEN'); // sufficient check
95          feeRateNumerator = _feeRateNumerator;
96          emit SetFeeRateNumerator(_feeRateNumerator);
97      }
```

Recommendation

We recommend adding reasonable upper and lower boundaries to all the configuration variables.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding in the commit 6f329818e575ce8736a28511598cdd9d8b7b28a3.

APPENDIX | LIGHTDAO II

Finding Categories

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

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 | CERTIK

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.

CertiK | **Securing** the **Web3** World

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.