CERTIK

Security Assessment

LightDAO

CertiK Verified on Mar 17th, 2023

CertiK Verified on Mar 17th, 2023

LightDAO

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

Executive Summary

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

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/17/2023	N/A

CODEBASE

https://github.com/Light-Ecosystem/light-lib/tree/cc564c955053e1c43132fbd0fe78826221a6a16f/contracts

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

...View All

COMMITS

cc564c955053e1c43132fbd0fe78826221a6a16f

1ac53e7dadd58b7f4c4fd4be4722285a58fc1db2

eb70734264b46194eb3f7f11335a01588298c3cc

...View All

Vulnerability Summary

15	12	1	0	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.
3	Major	1 Resolved, 2 Acknowledged	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.
2	Medium	1 Resolved, 1 Mitigated	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
7	Minor	7 Resolved	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.
3	Informational	3 Resolved	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

CODEBASE | LIGHTDAO

Repository

https://github.com/Light-Ecosystem/light-lib/tree/cc564c955053e1c43132fbd0fe78826221a6a16f/contracts
https://github.com/Light-Ecosystem/light-dao/commit/1ac53e7dadd58b7f4c4fd4be4722285a58fc1db2
https://github.com/Light-Ecosystem/permit2/commit/2141eef4c8b1ceb352fc48de441d7ea50fa2b967

Commit

cc564c955053e1c43132fbd0fe78826221a6a16f

1ac53e7dadd58b7f4c4fd4be4722285a58fc1db2

eb70734264b46194eb3f7f11335a01588298c3cc

2141eef4c8b1ceb352fc48de441d7ea50fa2b967

AUDIT SCOPE | LIGHTDAO

41 files audited ● 8 files with Acknowledged findings ● 30 files with Partially Resolved findings ● 3 files with Resolved findings

ID	File	SHA256 Checksum
AML	contracts/agents/AgentManager.sol	a106855b5994dffdfc9f9802813fb3d5324dcde0abc56e3c6ec0b2b6313b5300
HOP	contracts/agents/HOPESalesAgent.sol	5616ad3193b0bb4b9411a10d557de15b062056a90a35cc8f704e4179d813caf0
PGL	contracts/gombocs/PoolGomboc.sol	511fa22d902650552c2f2f5e26bf853fa5c1b90b69197f87ce12f3ead126a472
HOE	contracts/tokens/HOPE.sol	736d13035b152c748400028124e85a80c59bd57d86f7298dd610712beb98e90c
LTL	contracts/tokens/LT.sol	8b082bcb8a3d0c43c13391f2afba917e1a4e14c0c6d9a74812fad71f6143d40b
GCL	contracts/GombocController.sol	c2b3f721d2a0a360e11f17700eeb33c82786140137d6ec49c67d41db0cc07033
RLL	contracts/RestrictedList.sol	f4ac4a12940e2e06a74dfb931e0d5b8f29f464b774f8a098c22b86739a1a1a7e
VEL	contracts/VotingEscrow.sol	7a39ea37ce277a75b0306a2f14e2ff6ef98f41a76720df352690e1dc160ba1e0
AEL	contracts/gombocs/AbsExternalLTRewardDistributor.sol	822f977706fc2c402335ab22964ff34fcb0ae9c839ce02cf80e58d2b75fbde12
IAM	contracts/interfaces/IAgentManager.sol	5864d3b52e2238168cf86df8a12de3461a6ac30eb4f4f974929e512fb6d620be
IGC	contracts/interfaces/IGombocController.sol	b4a815d198b3d59f8359a4e32447abafcc9a21af7ef351975ae1ef1b7ea2e44e
IHO	contracts/interfaces/IHOPE.sol	9c5d91133e3d9547eac7f50aba9770995973392f9a51cc6c8dff40be954c12f4
IHP	contracts/interfaces/IHOPESalesAgent.sol	2186218528635efcd5685251427cff6384813dab2e7a19e884ed4d2df23631f5
ILT	contracts/interfaces/ILT.sol	0df1f34f38861da3bda34cd08448bef3d5d6efffe7913b375ecf75403f85eee

ID	File	SHA256 Checksum
IML	contracts/interfaces/IMinter.sol	6c65f8bfd655c206d149865b8da88b8c172aee1304916674f17b81959ac1a6e6
IPL	contracts/interfaces/IPermit2.sol	7fcb285395e3a899c174623f6f0c8642a9211779e16aac4d5204fd27fe081619
IRL	contracts/interfaces/IRestrictedList.sol	5d5ab47e1311e2e7120dba88a25a0fd299300cc2af97e7bce47b85a082746f3a
ISL	contracts/interfaces/IStaking.sol	ed3eb3cfacd3f1f1c29aa1e6b57f5f40777ae550d50a390d02dacba00a6fc85a
IVE	contracts/interfaces/IVotingEscrow.sol	121c9966b2ce6f18cd116346da9a42e32ceaa6776a0efceae4e68ad154042d9b
MLE	contracts/Minter.sol	5e530f3545bdc92ca267345a7da4574f1f53a7904afca1071b468252932e0355
IER	IERC20.sol	55d8c7efcfb6225ca2b7b36e04308f738bed1598128877efc517d97443601df3
IPE	IPermit2.sol	7fcb285395e3a899c174623f6f0c8642a9211779e16aac4d5204fd27fe081619
SUL	StringUtils.sol	61df1d3c89af46ccdf4c7d496369e55b7d9f60fa7317c02d23507dc25eb87654
THL	TransferHelper.sol	42a77d79639f1c37be35b83ab6e3960346d469f31138259961f41315be0626bf
IAT	src/interfaces/IAllowanceTransfer.sol	aeecef7ca9a72ee02f1ef8edfb44599339c5dc758cf382455c2f72fb77dffc74
IDA	src/interfaces/IDAIPermit.sol	13a98c0cbfa847ea57a8b427f76762b40de5fd10472e9d26e3aaa26de11e7c49
IEC	src/interfaces/IERC1271.sol	ba35907d098ef8b6c6c967d522026d09c92c7e2344cc81e4ee89db8955fa4ca8
IST	src/interfaces/ISignatureTransfer.sol	0a5d6ac59da350987693f40f0ff0a833f31b37e057645f8e6ed9a1dad691e45c
ALE	src/libraries/Allowance.sol	6c7d1edc74a9dca9940fa835db09302ed1a35019d69f4188351466e9a3fff458
PLL	src/libraries/Permit2Lib.sol	eabbad483496c0e4a5fe500b824d7f6c8bcc61a6edb49e1724c47d40d55838b8
PHL	src/libraries/PermitHash.sol	23f89633dbd02d8b52a33de3a915af5af73ff25e31f88b6cf994358f802efa60

ID	File	SHA256 Checksum
SCL	src/libraries/SafeCast160.sol	b106146964044fd9c89df48721011c43785982bf0dcb74a241f7a6f270fa8b2b
SVL	src/libraries/SignatureVerification.sol	1804b3d7b1183225419ec8ee45daf89174ff3668315d71d28111d5fcc179f8e4
ATL	src/AllowanceTransfer.sol	662e206052b8afec12f6270f1ec4e1d68723c8dc9e2dadfc1f2feed1dcb8c831
EIP	src/EIP712.sol	195ebab17589ed34e23de94eb9238bd099acaba88cc64b49c2763c72083bec98
PLE	src/Permit2.sol	a19dd81d4edafe3bba0178abfc9063886c2981b4711a4cbbf3fac19370defc9a
PEL	src/PermitErrors.sol	0919679c5bb58466d3a06d9d3297ec4946ba5d135330f10f4fb080067586afad
STL	src/SignatureTransfer.sol	234effebbe1e7ad7a3c791ea2b411f8e90993b758479817eb32c8767ea62ee8d
AGL	contracts/gombocs/AbsGomboc.sol	d1a30588c22439a49b6383aaaa032233880fab7af5b2c18dcbda9726d142223e
GFL	contracts/gombocs/GombocFactory.sol	2510073b02f9c843b8d75d46cc86fa212810b81bf8c709011a97ab19222516e6
SHO	contracts/StakingHOPE.sol	4e0b4c226b496925f9e9f3767a9201b64dd923948d8c9b7db7a080d1d9a1f745

APPROACH & METHODS | LIGHTDAO

This report has been prepared for LightDAO to discover issues and vulnerabilities in the source code of the LightDAO 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

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

System Overview

In the protocol, the user can buy the HOPE token with different assets, such as BTC, ETH, DAI, etc. Then the user can stake the HOPE tokens in the contract `StakingHOPE` and get the `stHOPE` token minted as a voucher. During the staking, the user can get the `LT` tokens as the staking rewards. The reward amount calculation logic is similar to the Curve protocol. After the staking, the user performs unstaking with the `stHOPE` token burnt and gets the `HOPE` token back.

While holding the `LT` tokens, the user can lock the `LT` tokens for voting power. 10000 `LT` token locked for four years equals 1 veLT weight. The voting can affect the pool weight and gain more `LT` tokens as boost rewards. The logic is also similar to the Curve protocol.

In addition, there are other staking pools where the user can stake other `lp` tokens. In these pools, the user can get additional rewards provided by different reward distributors.

Financial Models

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Notes

The permit2 repository in the codebase is forked from the Uniswap protocol:

https://github.com/Uniswap/permit2

Only the differences were reviewed. The audit scope only includes the delta part between these two protocols.

FINDINGS | LIGHTDAO

15	0	3	2	7	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for LightDAO. Through this audit, we have uncovered 15 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
LEB-01	State Variable Initialization In Constructor Within An Upgradeable Contract	Logical Issue	Major	Resolved
LEB-02	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
LET-01	**Centralized Control Of Contract Upgrade**	**Centralization / Privilege**	**Major**	**Acknowledged**
LEH-01	Different Transfer Implementation	Logical Issue	Medium	Resolved
LTL-01	**Initial Token Distribution**	**Centralization / Privilege**	**Medium**	**Mitigated**
AEL-01	Improper Variable Declarations	Coding Style	Minor	Resolved
GFL-01	Unknown Externally Owned Account(EOA)	Volatile Code	Minor	Resolved
GFL-02	Lack Of Access Control	Logical Issue	Minor	Resolved
LEB-03	Lack Of Storage Gap In Upgradeable Contract	Logical Issue	Minor	Resolved
LEB-04	Unused Return Value	Volatile Code	Minor	Resolved
LET-02	Unprotected Initializer	Control Flow	Minor	Resolved

ID	Title	Category	Severity	Status
SHO-01	Incorrect `require` Condition	Logical Issue	Minor	Resolved
HOP-01	Lack Of Price Oracle	Logical Issue	Informational	Resolved
HOP-02	Unable To Swap `Hope` For StableCoin	Logical Issue	Informational	Resolved
PGL-01	Missing Error Messages	Coding Style	Informational	Resolved

LEB-01 STATE VARIABLE INITIALIZATION IN CONSTRUCTOR WITHIN AN UPGRADEABLE CONTRACT

Category	Severity	Location	Status
Logical Issue	Major	contracts/StakingHOPE.sol (light-dao): 32~39; contracts/gombocs/PoolGomboc.sol (light-dao): 57~66	Resolved

Description

Code inside the constructor is only executed during the deployment of the contract and only affects the state of the implementation contract. Therefore, it will never be executed in the context of the proxy's state which makes it irrelevant for an upgradeable contract.

Recommendation

We recommend changing the contract's constructor into a regular function, typically named initialize, where you run all state variable initialization. And we advise calling `_disableInitialize` in the constructor to prevent the initializer from being called on the logic contract.

```
function initialize() public initializer {
     ... // state variable initialization
 }

/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
 _disableInitializers();
}
```

Alleviation

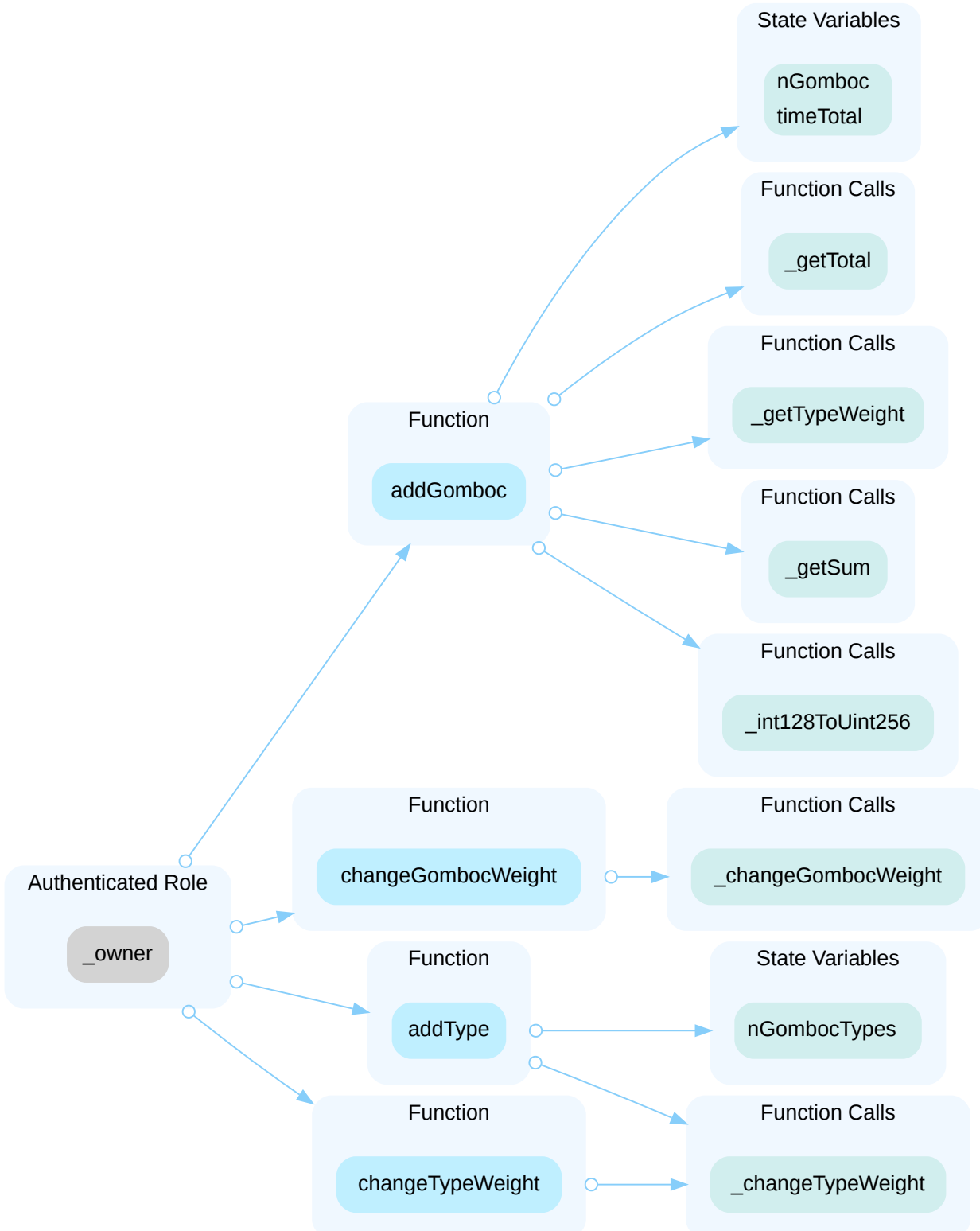
`[CertiK]`: The team heeded the advice and resolved the issue in the commit 8c6965ec84b5f68679c1e8c739e8a06a4c1947d3.

LEB-02 | CENTRALIZATION RELATED RISKS

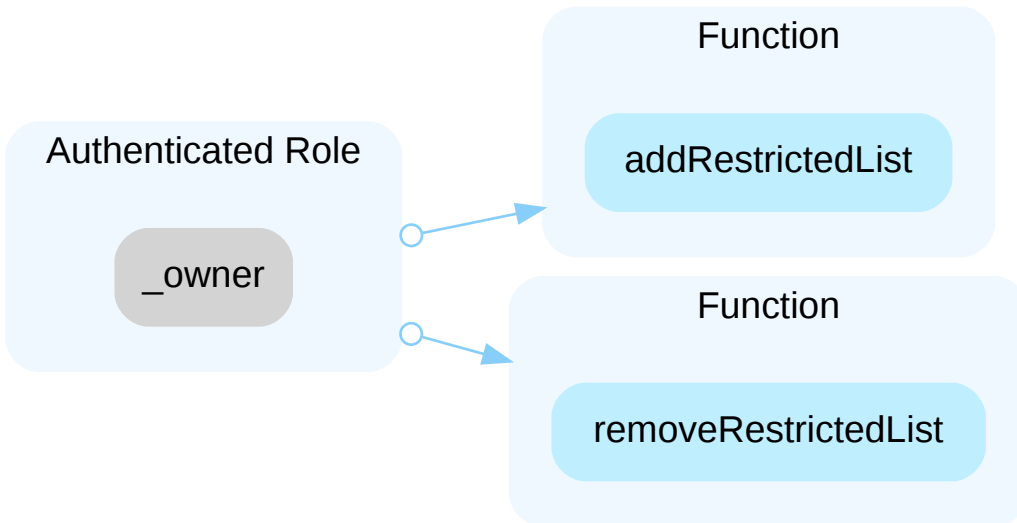
Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/GombocController.sol (light-dao); contracts/RestrictedList.sol (light-dao); contracts/VotingEscrow.sol (light-dao); contracts/agents/AgentManager.sol (light-dao); contracts/agents/HOPESalesAgent.sol (light-dao); contracts/gombocs/PoolGomboc.sol (light-dao); contracts/tokens/HOPE.sol (light-dao); contracts/tokens/LT.sol (light-dao)**	**Acknowledged**

Description

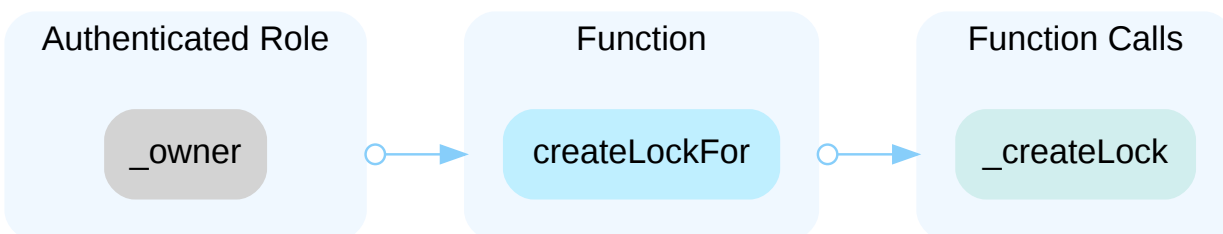
In the contract `GombocController` the role `_owner` has authority over the functions shown in the diagram below.



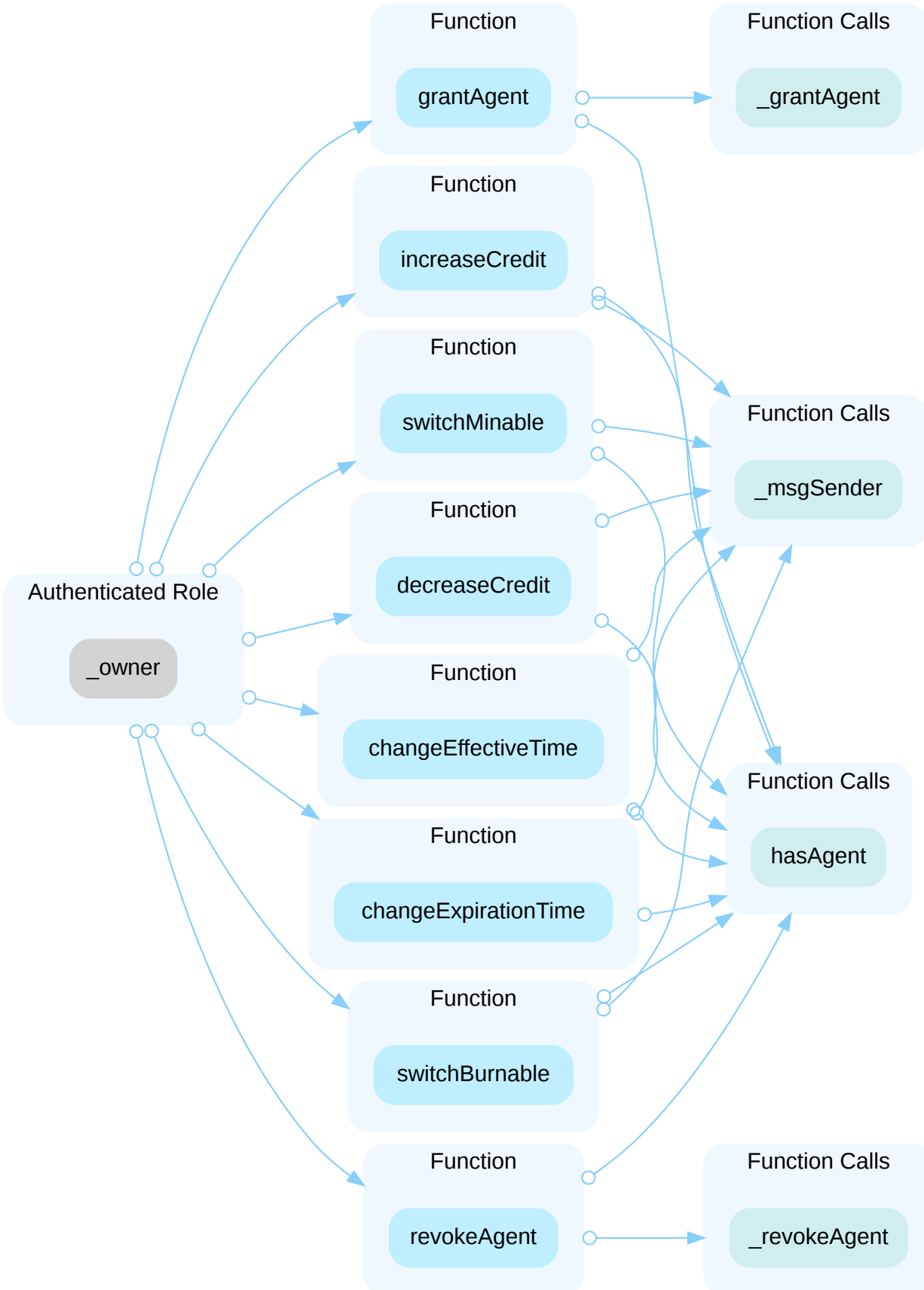
In the contract `RestrictedList` the role `_owner` has authority over the functions shown in the diagram below.



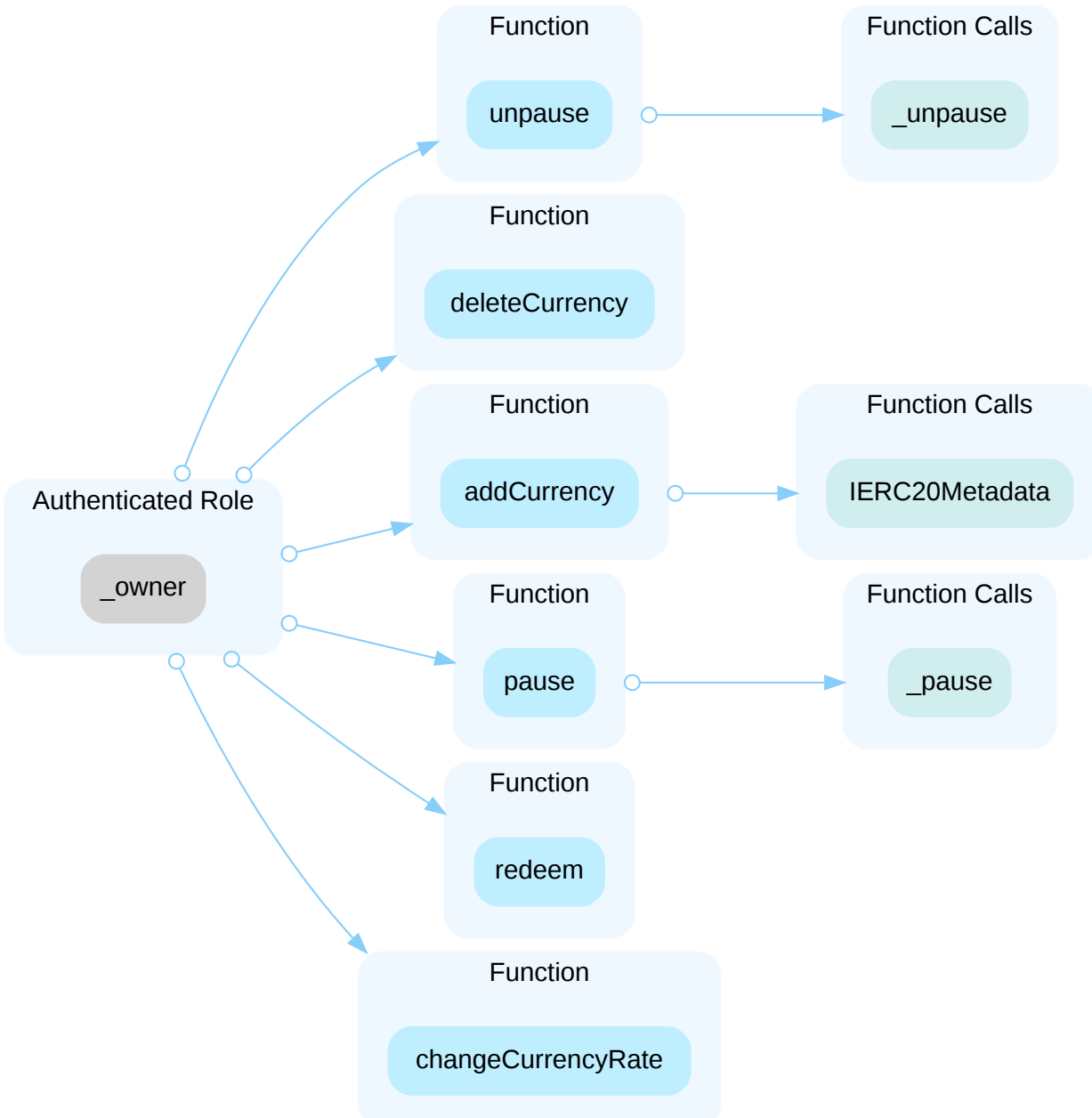
In the contract `VotingEscrow` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `AgentManager` the role `_owner` has authority over the functions shown in the diagram below.



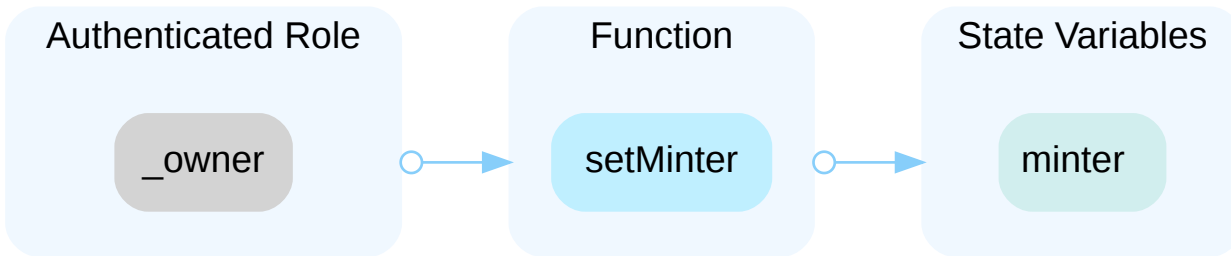
In the contract `HOPESalesAgent` the role `_owner` has authority over the functions shown in the diagram below.



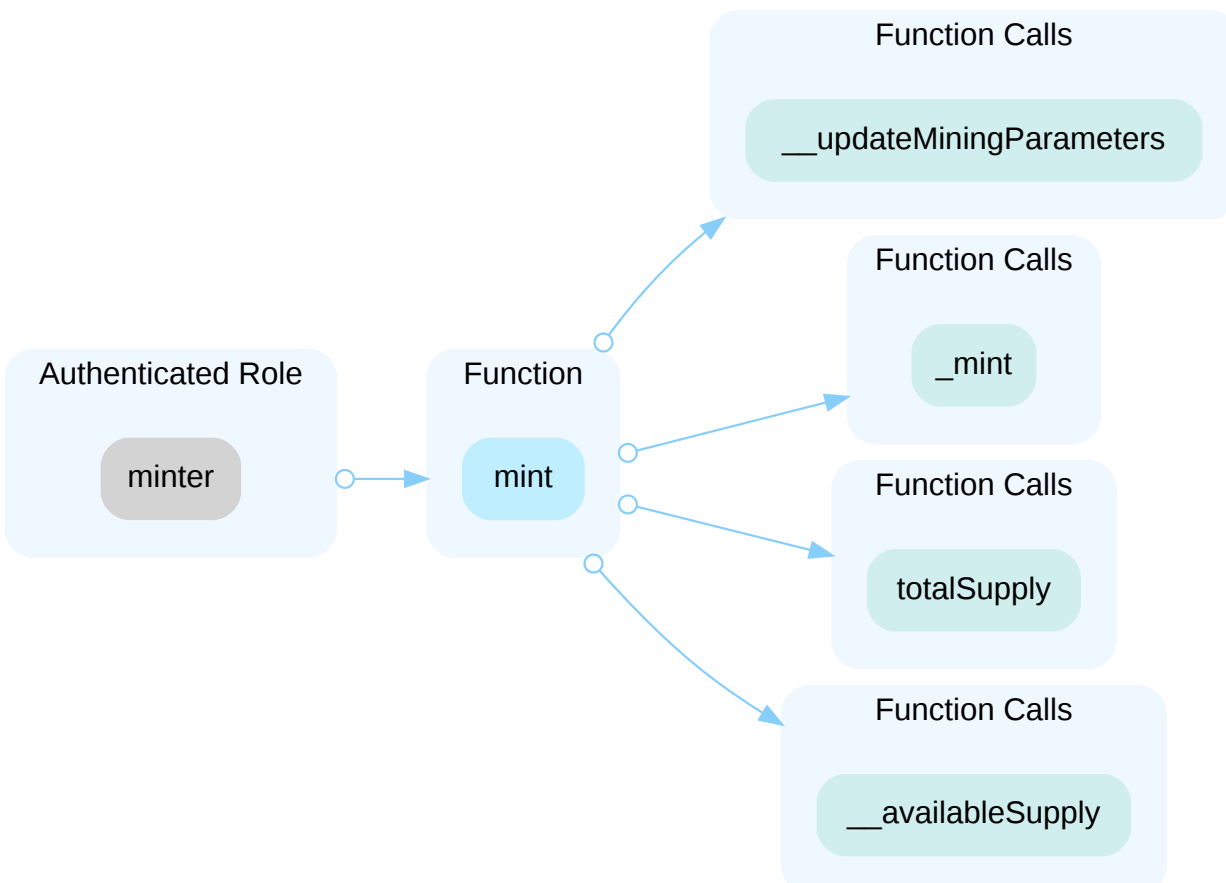
In the contract `PoolGomboc`, the role `_owner` has authority over the following functions:

- function addReward(): set up a new reward distributor contract.
- function setRewardDistributor(): change a reward distributor contract, also can be called by the current distributor contract itself.

In the contract `LT` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `LT` the role `minter` has authority over the functions shown in the diagram below.



In the contract `PoolGomboc` , the role `agent` has authority over the following functions:

- function mint(): mint `HOPE` tokens to an arbitrary address.
- function burn(): burn the `HOPE` tokens owned by the `agent` .

In the recent commit `9da9fda2705724115c76b32b5b38f588c5d30146` , a new function `setPermit2Address()` is introduced to the contracts `StakingHOPE` , `VotingEscrow` , `HOPESalesAgent` , `GombocFactory` and `PoolGomboc` . The role `_owner` can call this function to update the `Permit2` contract implementation.

Any compromise to the privileged account may allow the hacker to take advantage of this authority.

Recommendation

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 advise the client to carefully manage 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., multisignature 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.

Alleviation

`[LightDao]`: The governance/voting module is under development, and the ownership authority will be transferred to voting later.

LET-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/tokens/HOPE.sol (light-dao): 14; contracts/tokens/LT.sol (light-dao): 14**	**Acknowledged**

Description

`HOPE` and `LT` are upgradeable contracts, 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.

Recommendation

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 advise the client to carefully manage 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., multisignature 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.

Alleviation

`[LightDao]` : The governance/voting module is under development, and the ownership authority will be transferred to voting later.

LEH-01 | DIFFERENT TRANSFER IMPLEMENTATION

Category	Severity	Location	Status
Logical Issue	Medium	TransferHelper.sol (light-lib): 27, 64~86; src/SignatureTransfer.sol (permit2): 14, 123	Resolved

Description

The library `TransferHelper` uses different token transfer implementations for `doTransferIn()/doTransferInV2()` and `doTransferOut()`.

The function `doTransferIn()/doTransferInV2()` uses the library `SafeTransferLib` in `solmate` protocol. The transfer logic is shown below:

```
            success := and(
                // Set success to whether the call reverted, if not we check it
either
                // returned exactly 1 (can't just be non-zero data), or had no
return data.
                or(and(eq(mload(0), 1), gt(returndatasize(), 31)),
iszero(returndatasize())),
                // We use 68 because the length of our calldata totals up like so: 4
+ 32 * 2.
                // We use 0 and 32 to copy up to 32 bytes of return data into the
scratch space.
                // Counterintuitively, this call must be positioned second to the
or() call in the
                // surrounding and() call or else returndatasize() will be zero
during the computation.
                call(gas(), token, 0, freeMemoryPointer, 68, 0, 32)
            )
```

We can see that the condition `gt(returndatasize(), 31)` will allow any return value whose data size is greater than 31.

However, the transfer logic in the function `doTransferOut()` shown below will succeed only if the return data size is 0 or 32.

```
    function doTransferOut(address tokenAddress, address to, uint256 amount)
internal {
        IERC20 token = IERC20(tokenAddress);
        token.transfer(to, amount);

        bool success;
        assembly {
            switch returndatasize()
            case 0 {
                // This is a non-standard ERC-20
                success := not(0) // set success to true
            }
            case 32 {
                // This is a compliant ERC-20
                returndatacopy(0, 0, 32)
                success := mload(0) // Set `success = returndata` of override
external call
            }
            default {
                // This is an excessively non-compliant ERC-20, revert.
                revert(0, 0)
            }
        }
        require(success, "TOKEN_TRANSFER_OUT_FAILED");
    }
```

This will cause incompatibility with supported tokens. Since the `TransferHelper` is a library for extensibility, this incompatibility may result in potential bugs.

Recommendation

We recommend using same transfer implementation in a library.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit 51df9dc6a68069c04c09afcdcc584dcde28f166a.

LTL-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	**contracts/tokens/LT.sol (light-dao): 57**	**Mitigated**

Description

400_000_000_000 `LT` tokens are sent to the contract deployer when the contract is deployed. However, as described in the white paper, these tokens should be distributed to the relevant beneficiaries and locked for 4 years.

Recommendation

We recommend reviewing the logic and ensuring it is as intended.

Alleviation

`[LightDao]`:

The token distribution plan is published in the whitepaper:

https://hope.docsend.com/view/t6dpji9vj6rbz9mi

AEL-01 | IMPROPER VARIABLE DECLARATIONS

Category	Severity	Location	Status
Coding Style	Minor	contracts/gombocs/AbsExternalLTRewardDistributor.sol (light-dao): 17~19	Resolved

Description

The above-mentioned abstract contract declares the variables `_STHOPE GOMBOC` , `_MINTER` , and `_LTTOKEN` as `constant` and `private` , and initializes them to `address(0)` . This makes it impossible to access and change the values later in the derived contracts.

```
17      address private constant _STHOPE_GOMBOC = address(0);
18      address private constant _MINTER = address(0);
19      address private constant _LTToken = address(0);
```

Recommendation

We recommending properly declaring and initializing variables.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the issue in the commit 1792b98f9bd7e1d4f1ff06ef0916b7d394c6ca03.

GFL-01 | UNKNOWN EXTERNALLY OWNED ACCOUNT(EOA)

Category	Severity	Location	Status
Volatile Code	Minor	contracts/gombocs/GombocFactory.sol (light-dao): 11~13	Resolved

Description

In the contract `GombocFactory`, the state variable `_MINTER` and `_PERMIT2_ADDRESS` are initialized as `0x393B2E10bdB74E3A20F410C0896cC5EBa7312EED` and `0x7b230b9d46dCC38dfbfc2ca3E89655166704f808`. The ownership of them is uncertain.

Recommendation

We recommend ensuring that the EOA addresses are correct and properly managed. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multi-signature wallets.

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

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

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit 8c6965ec84b5f68679c1e8c739e8a06a4c1947d3.

GFL-02 | LACK OF ACCESS CONTROL

Category	Severity	Location	Status
Logical Issue	Minor	contracts/gombocs/GombocFactory.sol (light-dao): 23	Resolved

Description

The function `deploy()` can be called by anyone as it has no access restriction. This enables anyone to call this and create a `Gomboc` contract.

Recommendation

Consider adding a modifier to control who can create the `Gomboc`.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit ab69fbb0f1fdcdb0bd0a1d961f32a724df72b540.

LEB-03 | LACK OF STORAGE GAP IN UPGRADEABLE CONTRACT

Category	Severity	Location	Status
Logical Issue	Minor	contracts/agents/AgentManager.sol (light-dao): 9; contracts/gombocs/AbsGomboc.sol (light-dao): 12	Resolved

Description

There is no storage gap preserved in the logic contract. Any logic contract that acts as a base contract that needs to be inherited by other upgradeable children should have a reasonable size of storage gap preserved for the new state variable introduced by the future upgrades.

Recommendation

We recommend having a storage gap of a reasonable size preserved in the logic contract in case that new state variables are introduced in future upgrades. For more information, please refer to: https://docs.openzeppelin.com/contracts/3.x/upgradeable#storage_gaps.

Alleviation

`[CertiK]` : The team heeded the advice and resolved the issue in the commit 00273ea43eb28645e0cfe0031d459c28dfa4f10d.

LEB-04 | UNUSED RETURN VALUE

Category	Severity	Location	Status
Volatile Code	Minor	contracts/Minter.sol (light-dao): 82, 88; contracts/gombocs/AbsExternalLT RewardDistributor.sol (light-dao): 33	Resolved

Description

The return value of an external call is not stored in a local or state variable.

```
82          LiquidityGomboc(gombocAddr).userCheckpoint(_for);
```

```
88              ILT(token).mint(_for, toMint);
```

```
33          IERC20(_LTToken).approve(_STHOPE_GOMBOC, claimableTokens);
```

Recommendation

We recommend checking or using the return values of all external function calls.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit df61a807b7d600e89de1d34f29de1c6abb6ccca2.

LET-02 | UNPROTECTED INITIALIZER

Category	Severity	Location	Status
Control Flow	Minor	contracts/tokens/HOPE.sol (light-dao): 18; contracts/tokens/LT.sol (light-dao): 52	Resolved

Description

One or more logic contracts do not protect their initializers. An attacker can call the initializer and assume ownership of the logic contract, whereby she can perform privileged operations that trick unsuspecting users into believing that she is the owner of the upgradeable contract.

Recommendation

We advise calling `_disableInitializers` in the constructor or giving the constructor the `initializer` modifier to prevent the initializer from being called on the logic contract.

Reference: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit f6ebbf4c5919df91339723c0fe9297c0cbe5e741.

SHO-01 | INCORRECT `require` CONDITION

Category	Severity	Location	Status
Logical Issue	Minor	contracts/StakingHOPE.sol (light-dao): 80	Resolved

Description

The `require` condition is not correct, unstaking LP balance is not considered.

```
80      uint256 balanceOfUser = balanceOf(staker);
81      require(balanceOfUser >= amount, "INVALID_AMOUNT");
```

Recommendation

We recommend using the `lpBalanceOf(staker)` instead of `balanceOf(staker)`.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit 20f7e2b374660eecfb18a6a21224aa538a964307.

HOP-01 | LACK OF PRICE ORACLE

Category	Severity	Location	Status
Logical Issue	Informational	contracts/agents/HOPESalesAgent.sol (light-dao)	Resolved

Description

The contract `HOPESalesAgent` is selling `HOPE` tokens for different ERC20 tokens set in the `Currency` struct. The price is also stored in the `Currency.rate` variable and is centrally set by the contract `owner`. This price may be decoupled from market prices. Malicious users can leverage this for arbitrage. In addition, central price updating in the middle of the selling may cause the previous buyer losses.

Recommendation

We recommend obtaining the market price from Price Oracle.

Alleviation

`[LightDAO]`: The fundraising period will last only 24 hours, therefore the price just needs to be set once.

HOP-02 | UNABLE TO SWAP `Hope` FOR STABLECOIN

Category	Severity	Location	Status
Logical Issue	Informational	contracts/agents/HOPESalesAgent.sol (light-dao): 18	Resolved

Description

Users can buy `Hope` tokens through the `HOPESalesAgent` contract and then use those tokens to make staking on the minting of `LT` tokens. When users want to quit the market, they cannot swap `Hope` tokens for stablecoins. However, there is a function `redeem()` in the contract that allows the contract owner to withdraw all stablecoins. We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We recommend reviewing the logic again and ensuring it is as intended.

Alleviation

`[LightDao]`: The first phase of the project is designed in such a way that it does not provide the functionality to burn HOPE for retrieving stablecoin tokens. In the second phase of the project, the AgentManager adds the specified exchange as Agent, and grants the Agent burnable permission, so that the user can burn HOPE to retrieve stablecoin tokens through the Agent.

PGL-01 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/gombocs/PoolGomboc.sol (light-dao): 273, 282, 283, 284, 293, 358, 416	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit 05ee56f665d142a68ec9cc4b3a668ef59bd2213d.

OPTIMIZATIONS | LIGHTDAO

ID	Title	Category	Severity	Status
AGL-01	Floor Operation Can Be Taken Into The Library	Mathematical Operations	Optimization	Resolved
GCL-01	Using Dynamic Arrays	Gas Optimization	Optimization	Resolved
LEB-05	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Resolved
LEH-02	Solidity Version Not Recommended	Language Specific	Optimization	Partially Resolved

AGL-01 | FLOOR OPERATION CAN BE TAKEN INTO THE LIBRARY

Category	Severity	Location	Status
Mathematical Operations	Optimization	contracts/gombocs/AbsGomboc.sol (light-dao): 168	Resolved

Description

There are lots of `floor` calculations in the protocol. They are better to be put into a library to improve code readability.

Scenario

For example, the linked code is redundant, because the function `controller.gombocRelativeWeight()` in the contract `GombocController` will perform the `floor` calculation on the `time` variable again.

```
488  uint256 t = (time / _WEEK) * _WEEK;
```

Recommendation

We recommend adding a library to perform common mathematic calculations such as `floor`. The `prb-math` protocol can be a reference: https://github.com/PaulRBerg/prb-math/blob/main/src/Common.sol.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the finding in the commits 21e4528a699a1367ca727158c473a1029bc858e2 and 05962c5177666e32dcf1a985f82acb67212bae0d.

GCL-01 | USING DYNAMIC ARRAYS

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/GombocController.sol (light-dao): 30, 61, 71	Resolved

Description

The arrays declared in the contract `GombocController` have a very large compile-time fixed size. This will consume a large storage size.

Recommendation

We recommend using dynamic array instead.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the issue in the commit 1fd9dcdb14f39193631f87ecd593321260be4a5e.

LEB-05 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/GombocController.sol (light-dao): 19, 21; contracts/Minter.sol (light-dao): 17, 18; contracts/StakingHOPE.sol (light-dao): 12, 14; contracts/VotingEscrow.sol (light-dao): 56, 58, 73; contracts/agents/HOPESalesAgent.sol (light-dao): 29, 32, 35	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the finding in the commit 0c9fbfa79be8ed4284dfa55aa1c416eea3703754

LEH-02 | SOLIDITY VERSION NOT RECOMMENDED

Category	Severity	Location	Status
Language Specific	Optimization	contracts/Minter.sol (light-dao): 3; contracts/gombocs/AbsExternalLTRewardDistributor.sol (light-dao): 2; contracts/interfaces/IAgentManager.sol (light-dao): 3; contracts/interfaces/IGombocController.sol (light-dao): 3; contracts/interfaces/IHOPE.sol (light-dao): 3; contracts/interfaces/IHOPESalesAgent.sol (light-dao): 3; contracts/interfaces/ILT.sol (light-dao): 3; contracts/interfaces/IMinter.sol (light-dao): 2; contracts/interfaces/IPermit2.sol (light-dao): 3; contracts/interfaces/IRestrictedList.sol (light-dao): 3; contracts/interfaces/IStaking.sol (light-dao): 2; contracts/interfaces/IVotingEscrow.sol (light-dao): 2; IERC20.sol (light-lib): 4; IPermit2.sol (light-lib): 3; StringUtils.sol (light-lib): 3; TransferHelper.sol (light-lib): 3; src/AllowanceTransfer.sol (permit2): 2; src/EIP712.sol (permit2): 2; src/Permit2.sol (permit2): 2; src/PermitErrors.sol (permit2): 2; src/SignatureTransfer.sol (permit2): 2; src/interfaces/IAllowanceTransfer.sol (permit2): 2; src/interfaces/IDAIPermit.sol (permit2): 2; src/interfaces/IERC1271.sol (permit2): 2; src/interfaces/ISignatureTransfer.sol (permit2): 2; src/libraries/Allowance.sol (permit2): 2; src/libraries/Permit2Lib.sol (permit2): 2; src/libraries/PermitHash.sol (permit2): 2; src/libraries/SafeCast160.sol (permit2): 2; src/libraries/SignatureVerification.sol (permit2): 2	Partially Resolved

Description

Solidity frequently releases new compiler versions. Using an old version prevents access to new Solidity security features. We also recommend avoiding complex `pragma` statements which 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 recommend deploying 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

- 0.8.16

The recommendations take into account:

- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions. But, consider using the latest version of Solidity for testing.

Alleviation

`[CertiK]` : The team updated the code in commit eb805334c0c16bf42c866342ba50032218fc31f5 and partially resolved this issue.

APPENDIX | LIGHTDAO

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Mathematical Operations	Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

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

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

DISCLAIMER | 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.