Security Assessment

AMARA FINANCE

Jan 4th, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- GLOBAL-01 : Unlocked compiler version declaration
- GLOBAL-02 : Missing input validation
- GLOBAL-03 : Incorrect naming convention utilization
- GLOBAL-04 : Proper usage of “public” and “external” type
- GLOBAL-05 : Centralization risk
- AGA-01 : Centralization risk
- AGF-01 : Initial token distribution
- ALT-01 : Redundant code
- ATA-01 : Checks-Effects-Interactions pattern violations
- ATA-02 : Logical issue of function `exchangeRateStoredInternal()`
- CAA-01 : Misuse of a boolean constant
- CAA-02 : Centralization risk
- CAA-03 : Return value not stored
- CAA-04 : Centralization risk
- CAA-05 : Potential `mint/redeem/seize/transfer` failure possible
- GAF-01 : Lack of function `getPriorVotes()`
- SPO-01 : Centralization risk
- SPO-02 : Price oracle feed
- SPO-03 : Third party dependencies
- WGA-01 : Centralization risk

Appendix

Disclaimer

About

Summary

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

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	AMARA FINANCE
Description	AMARA FINANCE
Platform	moonriver
Language	Solidity
Codebase	https://github.com/AmaraFinance/Amara/tree/main/contracts
Commit	41628aabb12ed5d4bbbcd832f1c5b82d39d8aaeb

Audit Summary

Delivery Date	Jan 04, 2022
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	10	0	0	1	1	8
Medium	1	0	0	1	0	0
Minor	3	0	0	3	0	0
Informational	6	0	0	6	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
AGA	Governance/Amara.sol	632c47b90afa59fcfded603df11e5dbdb5a578d23d794229e5afcd2d50ab80c6
CGA	Governance/Context.sol	efccd8e1941e360829d21e1586fd204cb4e9b561ee940846fcf044fae829b607
GAG	Governance/GovernorAlpha.sol	9ef8938bd5c39c5f8f10fc7b3b357fc5b771dcded871c653fb8db34b5b413691
OGA	Governance/Ownable.sol	52d36e8031ba6ac07fa82c4ab5a880c5b2a5913ccfe7d66f059dac5d75957e2c
WGA	Governance/Whitelist.sol	766a8d83298fd0635bc1ba918b9e3b50f0dadd8fd64fcda85e5ae0635e1288f0
ALL	lens/AmaraLendLens.sol	48ef96763629b458f9a968052824b238723ae2ef15988d074d2b3d3e18b96a2c
ALT	lens/AmaraLendLensTool.sol	ed31f471862218d7531e3bd4a75e83ddda61dcdde13f0f850ee5ee6c75031af2
AEA	AErc20.sol	aa1acdca7c1011e99de9fb493cf3af3f9dca9892c089c778c0a0130d02d32708
AED	AErc20Delegate.sol	b052a092d32a144098ff48a5964a2afebb2de1e11df23743b94bfb89bfaa0a62
AEF	AErc20Delegator.sol	84421173aec150eff501c5552337a548972655721fc63b483e173331a838c32b
AEI	AErc20Immutable.sol	a3fa803b725c4778c9c2b72c36791af15133fdf4b31460e47512d55f80e1cc4f
ATA	AToken.sol	29200eb958a734fe754768ff8e4f34e819b0ab3a5029bcd41c6c3d7f9dda5cac
ATI	ATokenInterfaces.sol	1447bf26992341db382b7760526d4f9efde9a036fbccf8d337a2a167ade2c257
BJR	BaseJumpRateModelV2.sol	a3a08214ea5ebc1b3b74c8d8c500c36c1b145f67d5efb1a17ea4ffaacb38cb0c

ID	File	SHA256 Checksum
CMA	CarefulMath.sol	ee29c05e2fb79e4517ed8626e4819a0749937ad05f02c8292eabedf9a8b90b32
CAA	Comptroller.sol	634910b5d4889f9732173bcf05f479051daf9e62b349efa955874e69af2065c7
CIA	ComptrollerInterface.sol	58e13de723a6900348f08a30aaf1b11dc40352a3328aa706baa0d933eb16ee01
CSA	ComptrollerStorage.sol	77aa45fbaf65f976958e22fa55fb1f589c48592c925bcd11069622b96cdeace5
EIP	EIP20Interface.sol	7fbbb72ab5ab105ee9468d4330ce4f9a51d93e6b0fac9aeaf44700d0cc846abc
EIN	EIP20NonStandardInterface.sol	918d5790253d16e1b5221918d040399ad3598aec848b6a9007428965fe57e058
ERA	ErrorReporter.sol	b8a2e67f14e1fdc7b1eaa2df734a23497220ea5051cf2ffae97456785b2658bf
EAA	Exponential.sol	131dd94baf95d176d068e3fac92bcf93b07934efba8aff303f2dd626e4bc5110
ENE	ExponentialNoError.sol	3986c5175fbc1e6062ea4ee6603a80e63741dc05a98f6df40fd5810c1749770d
IRM	InterestRateModel.sol	dacc2f8a72c96904f25a8de2932db0894aac206e69d426a5ed3e69655bcf9aa8
JRM	JumpRateModel.sol	1cbf292da2f30223cd92e0b1db26e8690929a520445de979fd92adc5806ccf66
JRV	JumpRateModelV2.sol	953f080dcd24a6dba5066f3a42f606ead7c9a49706e5de493ce68b4f46365dcc
POA	PriceOracle.sol	18c2ac073559f9fcc644772ba3ef9b88eb447752e974a44f87259b2a6a45c391
RAA	Reservoir.sol	0ec5d36949ba40d88d92fa0b1d27382ac4457aa01244b7d8d3153ae9bf90eb2e
SMA	SafeMath.sol	204a19fb7a661c5bafcd5f7916254a457ca1fd9104e5708a73dd5010b11353dc

ID	File	SHA256 Checksum
SPO	SimplePriceOracleV2.sol	6fb726e615636f24ebe39f1872d10815bdce1392e17c34a01833cd3c5796e380
TAA	Timelock.sol	e583faa0fe824e3cbe65bbaaeb2b44be3a29a947faca2573474c782ee964fcf0
UAA	Unitroller.sol	fef99d5361d3629f643bd32cb8284b7f5dd7fce65fe4ab80039fa4429881909e
WPI	WhitePaperInterestRateModel.sol	fc1c4371e156b6a239b52e15eaa451d5eb19302e49e0636bde4fa8e7f33c7f10
AER	Governance/AERC20.sol	f56ffa75005da59218cc9b22922b595abc1ffd92dcf357bb2c7cdb49fc1a6246
AGF	Governance/Amara.sol	a79e5759f0753bbf8506c83d35a874c9369166e291a4949af10dc7c2d7528451
CGF	Governance/Context.sol	eb0167b1c14cef3031e76e798268da52fd19d43c30331f502f95bc5d5ad252f3
ERC	Governance/ERC20.sol	d04eac19f963c0b192df8fd1f7ac061222fd8dba502169d21fb5ea0890d3e3bd
GGA	Governance/GovernorAlpha.sol	9ef8938bd5c39c5f8f10fc7b3b357fc5b771dcded871c653fb8db34b5b413691
IER	Governance/IERC20.sol	05425b03777f63135cdae3494f276074a02cb5f924704e9f6a9249b84e13e23c
AAA	AEther.sol	2cbb8c215f49709a9a9e65956b49d224dc552e29c0b94bcd79e79db031ae1fe4

Findings

Critical	**0** (0.00%)
Major	**10** (50.00%)
Medium	**1** (5.00%)
Minor	**3** (15.00%)
Informational	**6** (30.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked compiler version declaration	Language Specific	Informational	Acknowledged
GLOBAL-02	Missing input validation	Volatile Code	Minor	Acknowledged
GLOBAL-03	Incorrect naming convention utilization	Coding Style	Informational	Acknowledged
GLOBAL-04	Proper usage of “public” and “external” type	Coding Style	Informational	Acknowledged
GLOBAL-05	Centralization risk	**Coding Style, Centralization / Privilege**	**Major**	Resolved
AGA-01	Centralization risk	**Centralization / Privilege**	**Major**	Resolved
AGF-01	Initial token distribution	**Centralization / Privilege**	**Medium**	Acknowledged
ALT-01	Redundant code	Logical Issue	Informational	Acknowledged
ATA-01	Checks-Effects-Interactions pattern violations	Logical Issue	Major	Resolved
ATA-02	Logical issue of function `exchangeRateStoredInternal()`	Logical Issue	Major	Partially Resolved

ID	Title	Category	Severity	Status
CAA-01	Misuse of a boolean constant	Coding Style	Informational	Acknowledged
CAA-02	Centralization risk	**Centralization / Privilege**	**Major**	Resolved
CAA-03	Return value not stored	Gas Optimization	Informational	Acknowledged
CAA-04	Centralization risk	**Centralization / Privilege**	**Major**	Resolved
CAA-05	Potential `mint/redeem/seize/transfer` failure possible	Logical Issue	Minor	Acknowledged
GAF-01	Lack of function `getPriorVotes()`	Logical Issue	Major	Resolved
SPO-01	Centralization risk	**Centralization / Privilege**	**Major**	Resolved
SPO-02	Price oracle feed	**Data Flow, Centralization / Privilege**	**Major**	Acknowledged
SPO-03	Third party dependencies	Volatile Code	Minor	Acknowledged
WGA-01	Centralization risk	**Centralization / Privilege**	**Major**	Resolved

GLOBAL-01 | Unlocked compiler version declaration

Category	Severity	Location	Status
Language Specific	Informational	Global	Acknowledged

Description

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting that a compiler version which is greater than the version will be used to compile the contracts.

Recommendation

It is a general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and thus be able to identify emerging more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GLOBAL-02 | Missing input validation

Category	Severity	Location	Status
Volatile Code	Minor	Global	Acknowledged

Description

The given input is missing the check for the non-zero address.

For example,

- contract `Comptroller`: `newBorrowCapGuardian` in function `_setBorrowCapGuardian()`, `newPauseGuardian` in function `_setPauseGuardian()`,
- contract `AToken`: `newPendingAdmin` in function `_setPendingAdmin()`,
- contract `Unitroller`: `newPendingImplementation` in function `_setPendingImplementation()`, `newPendingAdmin` in function `_setPendingAdmin()`

Recommendation

We recommend adding the check for the passed-in values to prevent unexpected error.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GLOBAL-03 | Incorrect naming convention utilization

Category	Severity	Location	Status
Coding Style	Informational	Global	Acknowledged

Description

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file:

Constants should be named with all capital letters with underscores separating words UPPER_CASE_WITH_UNDERSCORES

Functions other than constructors should use mixedCase

refer to https://solidity.readthedocs.io/en/v0.5.17/style-guide.html#naming-conventions

Examples:

Constants like :

- contract `ATokenStorage`: `borrowRateMaxMantissa`, `reserveFactorMaxMantissa`, `protocolSeizeShareMantissa`,
- contract `ATokenInterface`: `isAToken`,
- contract `Comptroller`: `compInitialIndex`, `closeFactorMinMantissa`, `closeFactorMaxMantissa`, `collateralFactorMaxMantissa`,
- contract `ComptrollerInterface`: `isComptroller`,
- contract `ExponentialNoError` `expScale`, `doubleScale`, `halfExpScale`, `mantissaOne`,
- contract `InterestRateModel`: `isInterestRateModel`,
- contract `PriceOracle`: `isPriceOracle`,

Functions like :

- contract `ExponentialNoError`: `mul_ScalarTruncate()`, `mul_ScalarTruncateAddUInt()`

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GLOBAL-04 | Proper usage of “public” and “external” type

Category	Severity	Location	Status
Coding Style	Informational	Global	Acknowledged

Description

“public” functions that are never called by the contract should be declared “external”. When the inputs are arrays, “external” functions are more efficient than “public” functions.

Examples:

Functions like :

- contract `Comptroller`: `enterMarkets()`, `getAccountLiquidity()`, `getHypotheticalAccountLiquidity()`, `_setPriceOracle()`, `_setPauseGuardian()`, `_setMintPaused()`, `_setBorrowPaused()`, `_setTransferPaused()`, `_setSeizePaused()`, `_become()`, `claimComp()`, `_grantComp()`, `_setCompSpeed()`, `_setContributorCompSpeed()`, `getAllMarkets()`, `setCompAddress()`,
- contract `AToken`: `initialize()`, `_setInterestRateModel()`,
- contract `Amara`: `delegate()`, `delegateBySig()`, `getPriorVotes()`
- contract `Unitroller`: `_setPendingImplementation()`, `_setPendingAdmin()`, `_acceptAdmin()`,
- contract `AErc20`: `initialize()`,
- contract `ACErc20Delegate`: `_becomeImplementation()`, `_resignImplementation()`,
- contract `AErc20Delegator`: `borrowBalanceStored()`, `exchangeRateCurrent()`, `exchangeRateStored()`, `accrueInterest()`, `_setComptroller()`, `_setInterestRateModel()`,

Recommendation

We recommend using the “external” attribute for functions never called from the contract.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GLOBAL-05 | Centralization risk

Category	Severity	Location	Status
Coding Style, Centralization / Privilege	Major	Global	Resolved

Description

In the contracts `AErc20Delegator/AToken/Unitroller`, the role `admin` has the authority over the following function:

- `_setImplementation()`: change the implementation of `AErc20` with any contracts,
- `_setComptroller()`: change the implementation of `Comptroller` with any contracts,
- `_setPendingImplementation()/_acceptImplementation()`: change the implementation of `Unitroller` with any contracts,
- `_setImplementation()`: change the `implementation` of `AmaraGovernanceDelegator` with any contracts,

Any compromise to the `admin` account may allow the hacker to take advantage of this and users' assets may suffer loss.

Recommendation

We advise the client to carefully manage the `admin` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or 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 the different levels in terms 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

The team acknowledged this issue and they transferred ownership to the Time-lock contract whose admin is a governance contract.

The contract `AMOVR` is deployed at
`https://moonriver.moonscan.io/address/0xf3f7cf36860CD5c912AB24ce5CD3adA46d7937d7`.

The contract `AUSDT` is deployed at
`https://moonriver.moonscan.io/address/0x2A58cd4D8fD217daa5530b8572B629a8cAE9a84A`.

The proxy of contract `Comptroller` is deployed at
`https://moonriver.moonscan.io/address/0x0bbB98dE6785127B34Ac458FE7B8be8DA34a68A3`.

The contract `Comptroller` is deployed at
`https://moonriver.moonscan.io/address/0x436285Be5FF0f69eA1ccE2aA8552690B7d9029Ae`.

The admin of contract `Comptroller/AMOVR/AUSDT` which is a timelock contract is deployed at
`https://moonriver.moonscan.io/address/0xddb86fd2E2d67d169377E3054EC5C5cF11a5f9E2`.

The admin of contract `Timelock` which is the proxy of governance contract is deployed at
`https://moonriver.moonscan.io/address/0xed6de6f6887af9e916b7f1410d2545dbdf084a8d`.

The implementation of the governance contract is deployed at
`https://moonriver.moonscan.io/address/0xb5baffe0dd4137d8d776147fa962be7097ee7f58`.

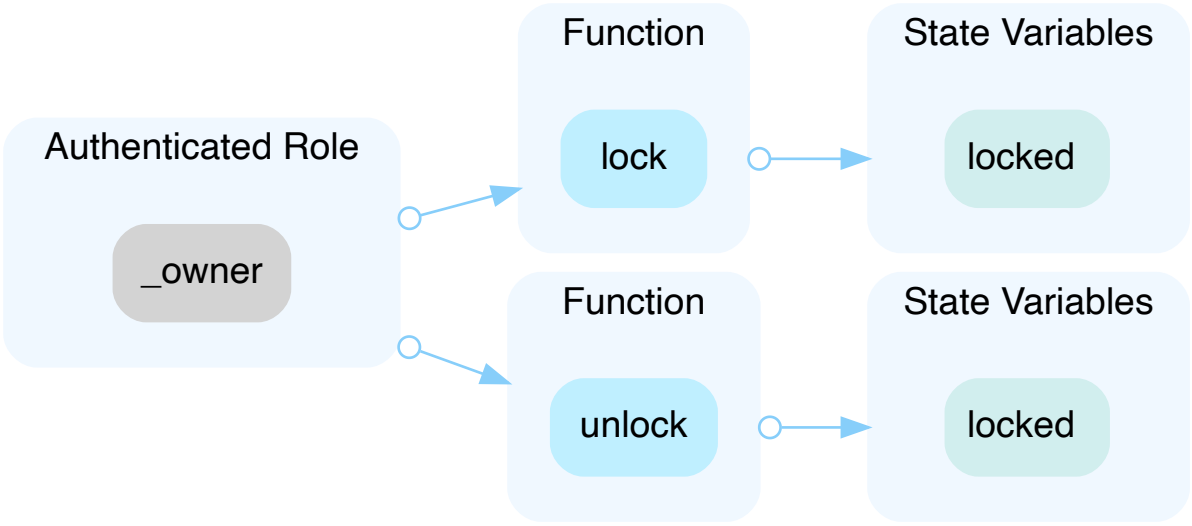
AGA-01 | Centralization risk

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/Governance/Amara.sol (41628aa): 311~313, 315~317	Resolved

Description

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

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this and users' assets may suffer loss.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature 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

The team heeded our advice and removed the previous contract `Amara` in commit `0d01878c8bca04ef057b5499cb7c76ed3d59bf5c`.

AGF-01 | Initial token distribution

Category	Severity	Location	Status
Centralization / Privilege	Medium	contracts/Governance/Amara.sol (6bcb6d8): 11	Acknowledged

Description

In the contract `Amara`, the deployer has the authority over the following function.

- `constructor()`: mint total tokens to the given addresses `_mintAddresses`

Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature 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

They team acknowledged this issue and they will mint tokens to multi-signature wallet when deploying.

ALT-01 | Redundant code

Category	Severity	Location	Status
Logical Issue	Informational	contracts/lens/AmaraLendLensTool.sol (41628aa): 85, 109	Acknowledged

Description

According to the following codes in the functions `getAccountBorrowAccrued()` and `getAccountSupplyAccrued()`, the checks `compBorrowerIndex == 0` and `compSupplierIndex == 0` are redundant.

```
 88          uint compBorrowerIndex = 0;
 89          uint224 borrowStateIndex;
 90
 91          Exp memory marketBorrowIndex = Exp({mantissa : aToken.borrowIndex()});
 92          if (compBorrowerIndex == 0) {
 93              compBorrowerIndex = comptroller.compBorrowerIndex(address(aToken),
account);
 94          }
```

```
112          uint compSupplierIndex = 0;
113          uint224 supplyStateIndex;
114
115          if (compSupplierIndex == 0) {
116              compSupplierIndex = comptroller.compSupplierIndex(address(aToken),
account);
117          }
```

Recommendation

We recommend removing the redundant codes.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

ATA-01 | Checks-Effects-Interactions pattern violations

Category	Severity	Location	Status
Logical Issue	Major	contracts/AToken.sol (41628aa): 695, 787	Resolved

Description

The following codes in the functions `redeemFresh()` and `borrowFresh()` do not meet the Checks-Effects-Interactions pattern.

```
695         doTransferOut(redeemer, vars.redeemAmount);
696
697         /* We write previously calculated values into storage */
698         totalSupply = vars.totalSupplyNew;
699         accountTokens[redeemer] = vars.accountTokensNew;
```

```
787         doTransferOut(borrower, borrowAmount);
788
789         /* We write the previously calculated values into storage */
790         accountBorrows[borrower].principal = vars.accountBorrowsNew;
791         accountBorrows[borrower].interestIndex = borrowIndex;
792         totalBorrows = vars.totalBorrowsNew;
```

It only has a reentrancy lock as there is no lock at the comptroller level, only the AToken level.

If the `aToken` is an ERC777 protocol, the reentrancy can happen in function levels of an ERC777 based contract, i.e. multiple function calls that are triggered by the hook mechanism of ERC777.

This issue is possible to happen with all compound forks, but Compound is not affected as they do not list tokens with callback functionality.

Recommendation

We recommend using the Checks-Effects-Interactions pattern and understanding the security limitations of forking compound.

Alleviation

The team heeded our advice and resolved this issue in commit `d26d095bc02e1ee812275102e0fef88e34fe9d45`.

ATA-02 | Logical issue of function `exchangeRateStoredInternal()`

Category	Severity	Location	Status
Logical Issue	Major	contracts/AToken.sol (41628aa): 340	Partially Resolved

Description

In the aforementioned line, the formula for the calculation of `exchangeRate` is as following after aToken is minted:

$$exchangeRate = \frac{totalCash + totalBorrows - totalReserves}{totalSupply}$$

```
340     function exchangeRateStoredInternal() internal view returns (MathError, uint) {
341         uint _totalSupply = totalSupply;
342         if (_totalSupply == 0) {
343             /*
344              * If there are no tokens minted:
345              *  exchangeRate = initialExchangeRate
346              */
347             return (MathError.NO_ERROR, initialExchangeRateMantissa);
348         } else {
349             /*
350              * Otherwise:
351              *  exchangeRate = (totalCash + totalBorrows - totalReserves) /
totalSupply
352              */
353             uint totalCash = getCashPrior();
354             uint cashPlusBorrowsMinusReserves;
355             Exp memory exchangeRate;
356             MathError mathErr;
357
358             (mathErr, cashPlusBorrowsMinusReserves) = addThenSubUInt(totalCash,
totalBorrows, totalReserves);
359             if (mathErr != MathError.NO_ERROR) {
360                 return (mathErr, 0);
361             }
362
363             (mathErr, exchangeRate) = getExp(cashPlusBorrowsMinusReserves,
_totalSupply);
364             if (mathErr != MathError.NO_ERROR) {
365                 return (mathErr, 0);
366             }
367
368             return (MathError.NO_ERROR, exchangeRate.mantissa);
```

```
369         }
370     }
```

In solidity, division calculations have truncation problems. The `totalSupply` will be 1 and `exchangeRate` will be much smaller than `initialExchangeRate` in case the last user redeems (`accountTokens[redeemer] - 1`) aToken.

As a result, the `exchangeRate` would be extremely small.

When the value of `exchangeRate` is much smaller than `initialExchangeRate`, the user can mint aTokens well above normal values, and then the value of `exchangeRate` will be normal with the interest generating. In other words, the users can use this arbitrage to take away the underlying tokens in this pool.

For example, the user can mint the amount of 1e8 aToken with one underlying token in case `exchangeRate = 1/1e8`.

Recommendation

We recommend using the following solutions to help mitigate this issue:

1. adding reasonable upper and lower boundaries to replace the return value when the `exchangeRate` is un-reasonable big or small.
2. adding a new contract that can only call `mint()` but can't call `redeem()` to supply reasonable amounts of the underlying token to the pool.

Alleviation

The team heeded our advice and added a new contract `LendMintProvider` that can only call `mint()` but can't call `redeem()` in commit `6c317d8bdfc2aa3e8386893f86c8b8de90ff6393`.

CAA-01 | Misuse of a boolean constant

Category	Severity	Location	Status
Coding Style	Informational	contracts/Comptroller.sol (41628aa)	Acknowledged

Description

Boolean constants in code have only a few legitimate uses. Other uses (in complex expressions, as conditionals) indicate either an error or, most likely, the persistence of faulty code.

Examples:

```
259         if (false) {
260             maxAssets = maxAssets;
261         }
```

Recommendation

We recommend removing the ineffectual code.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

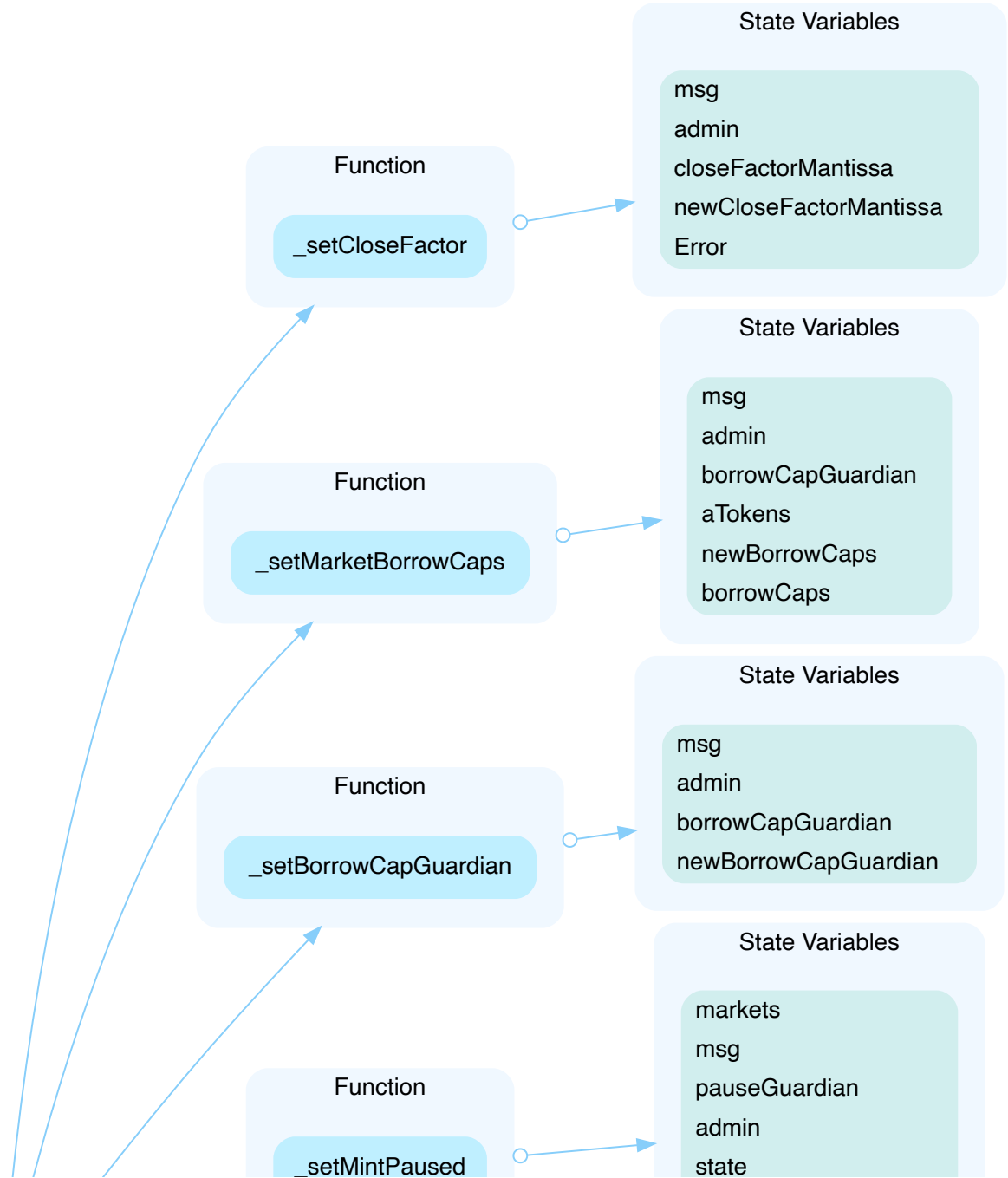
CAA-02 | Centralization risk

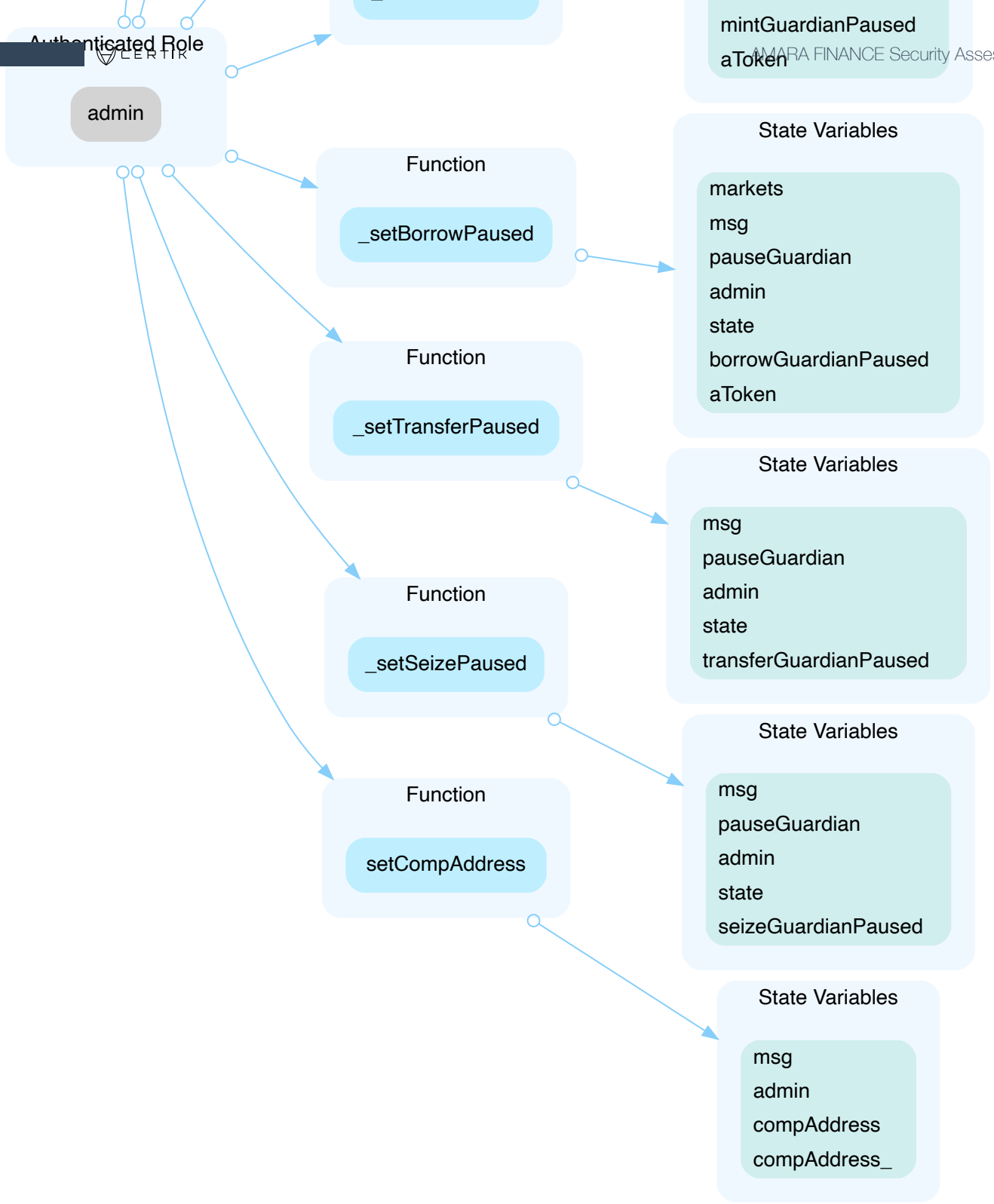
Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/Comptroller.sol (41628aa): 836~845, 955~967, 973~984, 1008~1016, 1018~1026, 1028~1035, 1037~1044, 1334~1337, 955~967, 1008~1016, 1018~1026, 1028~1035, 1037~1044	Resolved

Description

In the contract `Comptroller`, the role `admin` has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `admin` may allow the hacker to take advantage of this and users' assets may suffer loss.



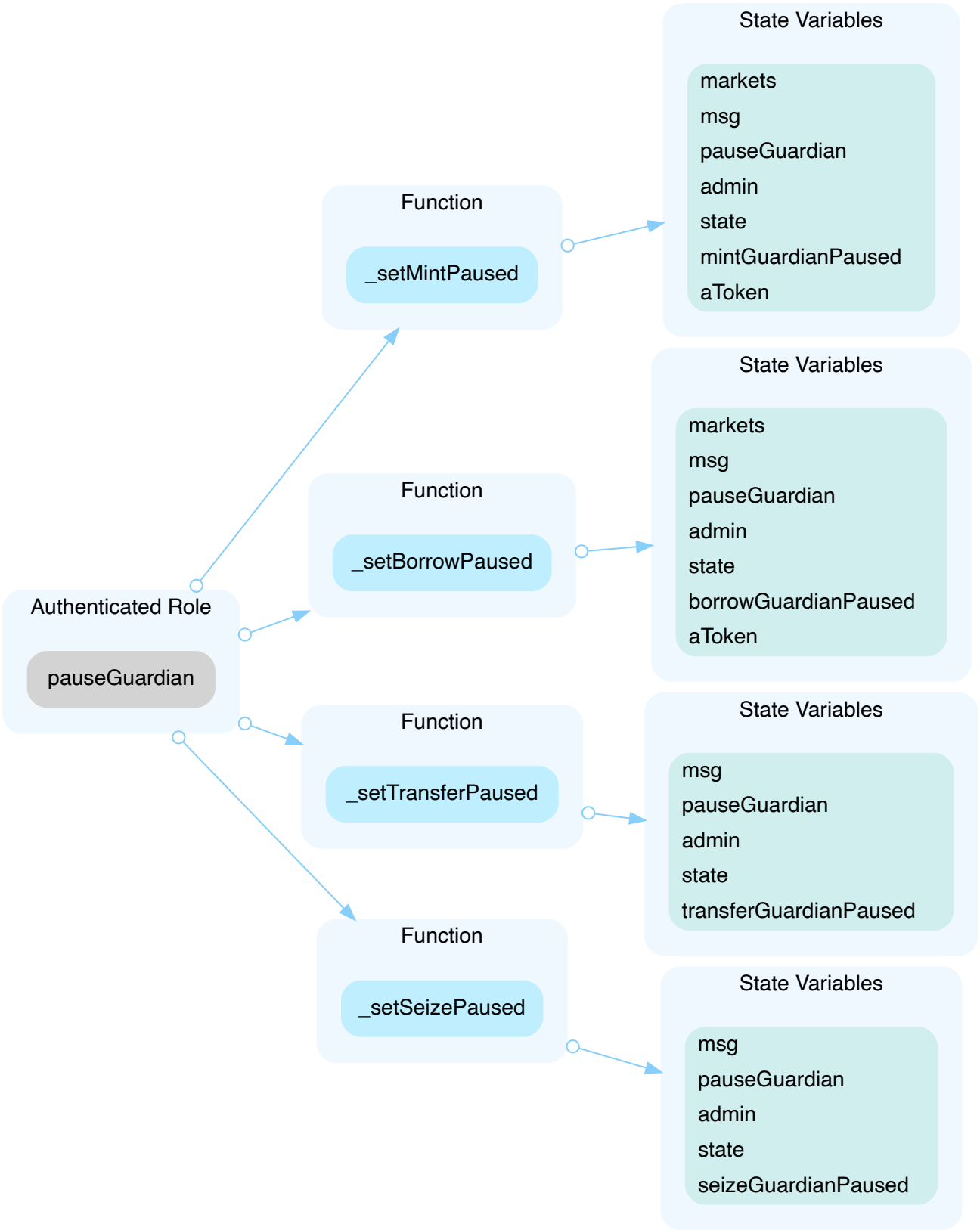


In the contract `Comptroller`, the role `borrowCapGuardian` has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `borrowCapGuardian` may allow the hacker to take advantage of this and users' assets may suffer loss.

In the contract `Comptroller`, the role `pauseGuardian` has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `pauseGuardian` may allow the hacker to take advantage of this and users' assets may suffer loss.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g.,

Multisignature 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

The team acknowledged this issue and they transferred ownership to the Time-lock contract whose admin is a governance contract.

The proxy of contract `Comptroller` is deployed at `https://moonriver.moonscan.io/address/0x0bbB98dE6785127B34Ac458FE7B8be8DA34a68A3`.

The contract `Comptroller` is deployed at `https://moonriver.moonscan.io/address/0x436285Be5FF0f69eA1ccE2aA8552690B7d9029Ae`.

The admin of contract `Comptroller` which is a timelock contract is deployed at `https://moonriver.moonscan.io/address/0xddb86fd2E2d67d169377E3054EC5C5cF11a5f9E2`.

The admin of contract `Timelock` which is the proxy of governance contract is deployed at `https://moonriver.moonscan.io/address/0xed6de6f6887af9e916b7f1410d2545dbdf084a8d`.

The implementation of the governance contract is deployed at `https://moonriver.moonscan.io/address/0xb5baffe0dd4137d8d776147fa962be7097ee7f58`.

CAA-03 | Return value not stored

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/Comptroller.sol (41628aa)	Acknowledged

Description

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

Examples:

```
function _supportMarket(AToken aToken) external returns (uint) {
  ...
  aToken.isAToken();
  ...
}
```

Recommendation

We recommend adding “require” statement for isAToken:

```
require(aToken.isAToken();,”This is not a AToken contract!”);
```

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

CAA-04 | Centralization risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/Comptroller.sol (41628aa): 1279	Resolved

Description

In the contract `Comptroller`, the role `admin` has the authority over the following function:

- `_grantComp()`: transfer Amara tokens to any addresses,

Any compromise to the `admin` account may allow the hacker to take advantage of this and users' assets may suffer loss.

Recommendation

We advise the client to carefully manage the `admin` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or 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 the different levels in terms 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

The team acknowledged this issue and they transferred ownership to the Time-lock contract whose admin is a governance contract.

The proxy of contract `Comptroller` is deployed at `https://moonriver.moonscan.io/address/0x0bbB98dE6785127B34Ac458FE7B8be8DA34a68A3`.

The contract `Comptroller` is deployed at `https://moonriver.moonscan.io/address/0x436285Be5FF0f69eA1ccE2aA8552690B7d9029Ae`.

The admin of contract `Comptroller` which is a timelock contract is deployed at `https://moonriver.moonscan.io/address/0xddb86fd2E2d67d169377E3054EC5C5cF11a5f9E2`.

The admin of contract `Timelock` which is the proxy of governance contract is deployed at `https://moonriver.moonscan.io/address/0xed6de6f6887af9e916b7f1410d2545dbdf084a8d`.

The implementation of the governance contract is deployed at `https://moonriver.moonscan.io/address/0xb5baffe0dd4137d8d776147fa962be7097ee7f58`.

CAA-05 | Potential `mint/redeem/seize/transfer` failure possible

Category	Severity	Location	Status
Logical Issue	Minor	contracts/Comptroller.sol (41628aa): 1102, 1149	Acknowledged

Description

According to the codes in the function `distributeSupplierComp()`, the function is used to calculate the amount of Amara that needs to distribute to the user. The amount is calculated by the `deltaIndex`, which is calculated by the block-related parameters `supplyIndex(compSupplyState[aToken].index)` and `supplierIndex`. `supplierIndex` may be the value of `compInitialIndex`.

```
1149      function distributeSupplierComp(address aToken, address supplier) internal {
1150          CompMarketState storage supplyState = compSupplyState[aToken];
1151          Double memory supplyIndex = Double({mantissa : supplyState.index});
1152          Double memory supplierIndex = Double({mantissa : compSupplierIndex[aToken]
[supplier]});
1153          compSupplierIndex[aToken][supplier] = supplyIndex.mantissa;
1154
1155          if (supplierIndex.mantissa == 0 && supplyIndex.mantissa > 0) {
1156              supplierIndex.mantissa = compInitialIndex;
1157          }
1158
1159          Double memory deltaIndex = sub_(supplyIndex, supplierIndex);
1160          uint supplierTokens = AToken(aToken).balanceOf(supplier);
1161          uint supplierDelta = mul_(supplierTokens, deltaIndex);
1162          uint supplierAccrued = add_(compAccrued[supplier], supplierDelta);
1163          compAccrued[supplier] = supplierAccrued;
1164          emit DistributedSupplierComp(AToken(aToken), supplier, supplierDelta,
supplyIndex.mantissa);
1165      }
```

According to the codes in the function `updateCompSupplyIndex()`, `compSupplyState[aToken].index` is calculated by the block and the `supplySpeed`, which may be smaller the value of `compInitialIndex` in case `compSupplyState[aToken]` is initialized incorrectly.

```
1102      function updateCompSupplyIndex(address aToken) internal {
1103          CompMarketState storage supplyState = compSupplyState[aToken];
1104          uint supplySpeed = compSpeeds[aToken];
1105          uint blockNumber = getBlockNumber();
1106          uint deltaBlocks = sub_(blockNumber, uint(supplyState.block));
1107          if (deltaBlocks > 0 && supplySpeed > 0) {
1108              uint supplyTokens = AToken(aToken).totalSupply();
```

```
1109              uint compAccrued = mul_(deltaBlocks, supplySpeed);
1110              Double memory ratio = supplyTokens > 0 ? fraction(compAccrued,
supplyTokens) : Double({mantissa : 0});
1111              Double memory index = add_(Double({mantissa : supplyState.index}),
ratio);
1112              compSupplyState[aToken] = CompMarketState({
1113              index : safe224(index.mantissa, "new index exceeds 224 bits"),
1114              block : safe32(blockNumber, "block number exceeds 32 bits")
1115              });
1116          } else if (deltaBlocks > 0) {
1117              supplyState.block = safe32(blockNumber, "block number exceeds 32
bits");
1118          }
1119      }
```

As a result, the function `distributeSupplierComp()` called in the functions `mintAllowed()/redeemAllowed()/seizeAllowed()/transferAllowed()` will fail as subtraction overflow may be caused when calculating `deltaIndex`.

Recommendation

We recommend initializing the `compSupplyState[aToken]` correctly when deploying.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GAF-01 | Lack of function `getPriorVotes()`

Category	Severity	Location	Status
Logical Issue	Major	contracts/Governance/GovernorAlpha.sol (6bcb6d8): 137, 207, 266, 336	Resolved

Description

The function `getPriorVotes()` can't be found in the contract `Amara`, which will make functions `propose()`, `cancel()`, `castVote()` and `castVoteBySig()` fail to call.

Recommendation

We recommend implementing this function before deployment.

Alleviation

The team acknowledged this issue and they removed `GovernorAlpha.sol` in commit `4ecdf16f38e7ffc85c0517f63b546a73907ae55b`.

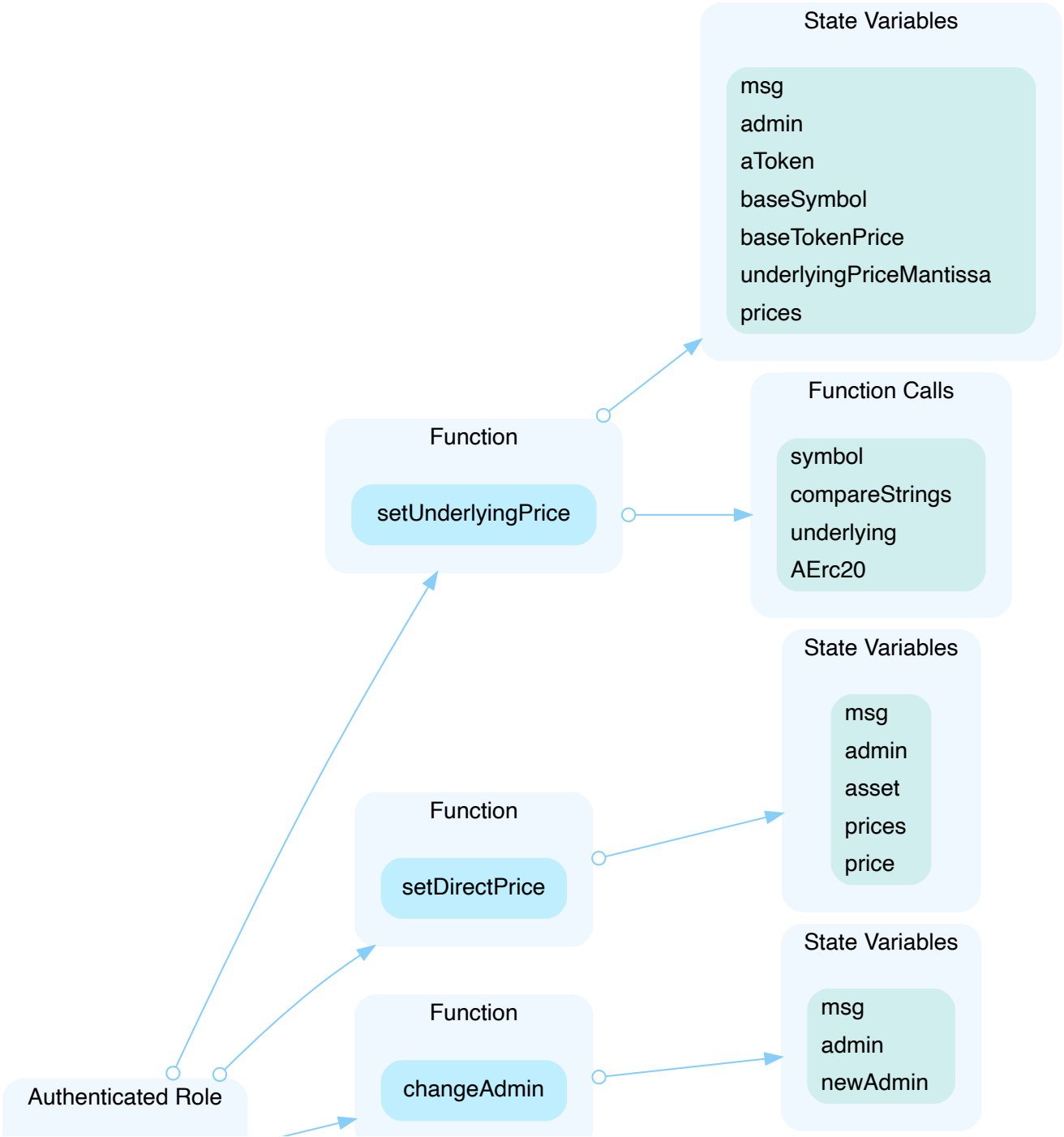
SPO-01 | Centralization risk

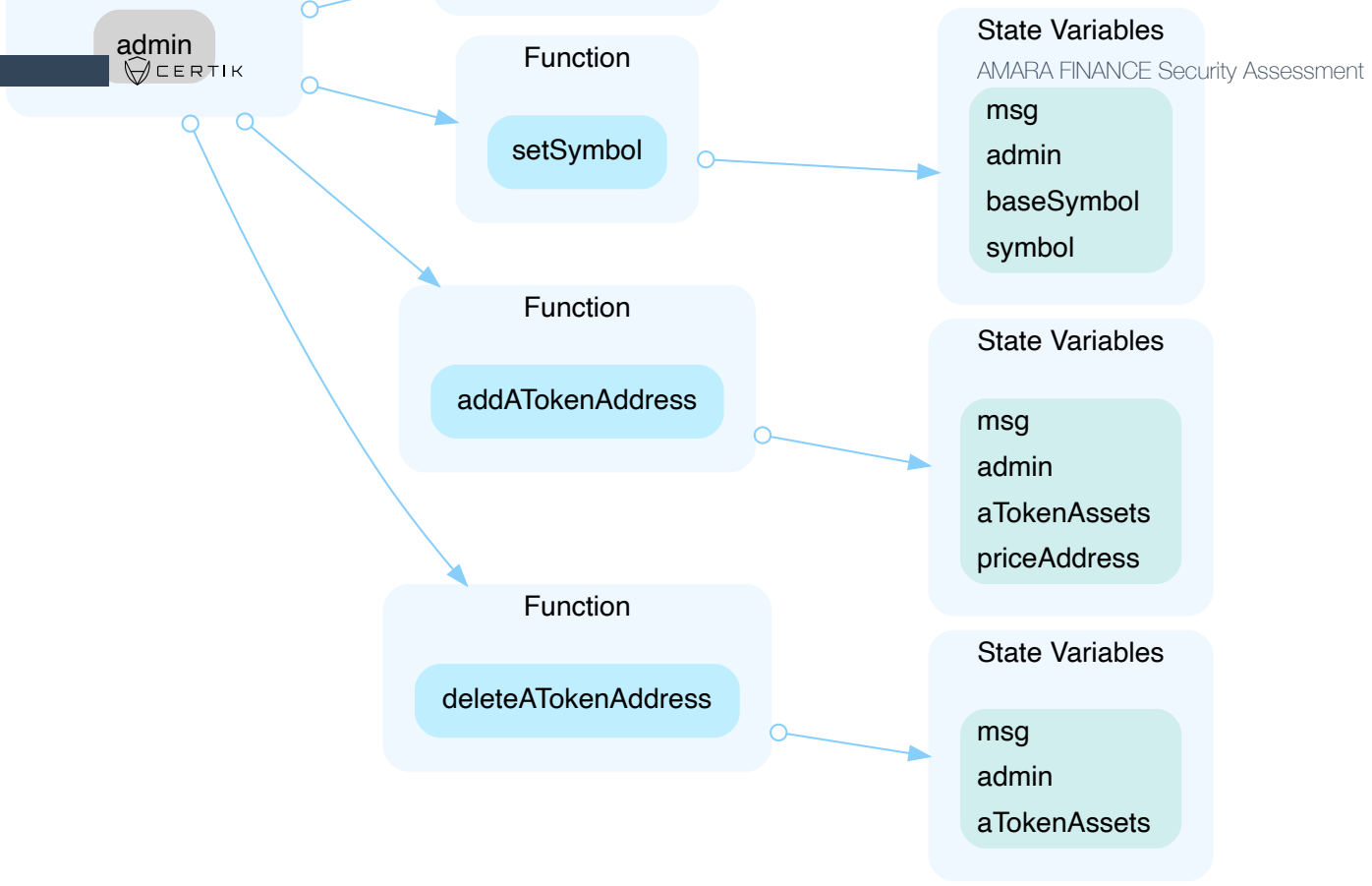
Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/SimplePriceOracleV2.sol (41628aa): 93~102, 104~108, 110~113, 125~128, 131~134, 136~139	Resolved

Description

In the contract `SimplePriceOracle`, the role `admin` has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `admin` may allow the hacker to take advantage of this and users' assets may suffer loss.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature 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

The team acknowledged this issue and they transferred ownership to the Time-lock contract whose admin is a governance contract.

The contract `SimplePriceOracle` is deployed at `https://moonriver.moonscan.io/address/0xF2938ab8655aA6D902b65A2F34Cc0e73e909a71f`.

The admin of contract `SimplePriceOracle` which is a timelock contract is deployed at `https://moonriver.moonscan.io/address/0xddb86fd2E2d67d169377E3054EC5C5cF11a5f9E2`.

The admin of contract `Timelock` which is the proxy of governance contract is deployed at `https://moonriver.moonscan.io/address/0xed6de6f6887af9e916b7f1410d2545dbdf084a8d`.

The implementation of the governance contract is deployed at `https://moonriver.moonscan.io/address/0xb5baffe0dd4137d8d776147fa962be7097ee7f58`.

SPO-02 | Price oracle feed

Category	Severity	Location	Status
Data Flow, Centralization / Privilege	**Major**	contracts/SimplePriceOracleV2.sol (41628aa)	Acknowledged

Description

A serious issue was caused by Compound’s centralized oracle solution which pulls market data from only a single exchange, Coinbase, with Uniswap TWAP used as a backstop.

Using Uniswap TWAP as a backstop is better than no backstop in this situation, but it introduces a false sense of security as it too can trivially be manipulated (as we saw during this event).

Recommendation

We recommend using reliable on-chain price oracle, such as Chainlink and Band protocol.

Alleviation

The team acknowledged this issue and they stated:

"They will use Chainlink as the price oracle feed."

SPO-03 | Third party dependencies

Category	Severity	Location	Status
Volatile Code	Minor	contracts/SimplePriceOracleV2.sol (41628aa): 80	Acknowledged

Description

The contract is serving as the underlying entity to interact with third party `ConsumerV3` protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of oracle requires interaction with `ConsumerV3`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

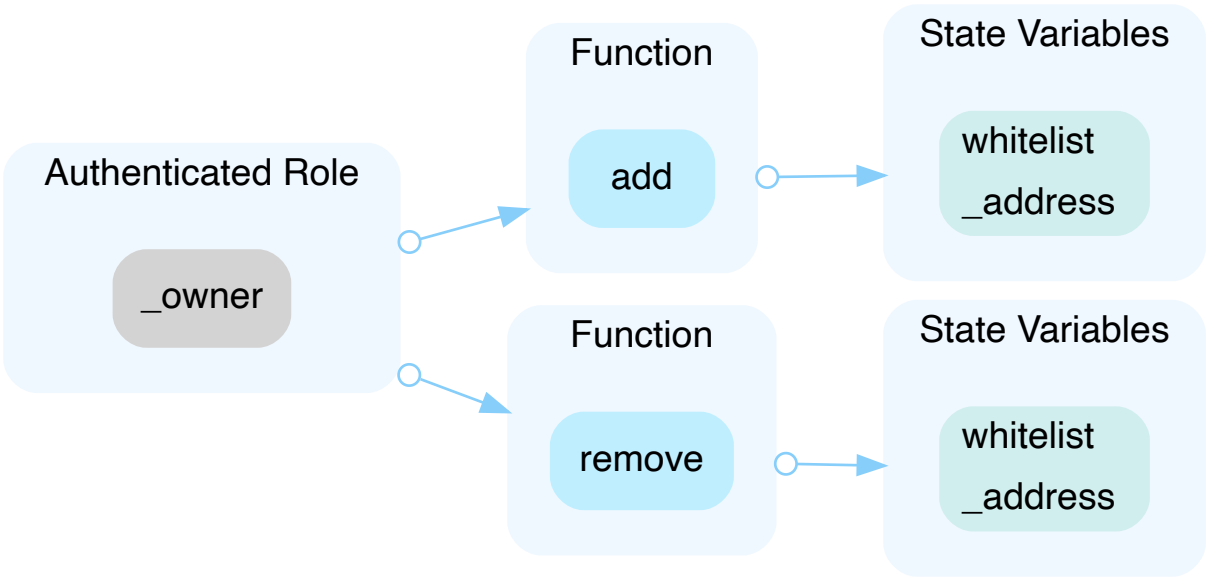
WGA-01 | Centralization risk

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/Governance/Whitelist.sol (41628aa): 16~19, 21~24	Resolved

Description

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

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this and users' assets may suffer loss.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature 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

The team heeded our advice and removed the contract `Whitelist` in commit `0d01878c8bca04ef057b5499cb7c76ed3d59bf5c`.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

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

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

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

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

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.