Security Assessment

Amara Finance II

Mar 31st, 2022

Table of Contents

Summary

This report has been prepared for Amara Finance II to discover issues and vulnerabilities in the source code of the Amara Finance II 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 II
Platform	Other
Language	Solidity
Codebase	https://github.com/xizho10/mara-auto/tree/main/contracts
Commit	6310fa663347c7656a6c5a1e4da27a0bcfd6cf92

Audit Summary

Delivery Date	Mar 31, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	5	0	0	4	0	0	1
Medium	0	0	0	0	0	0	0
Minor	1	0	0	1	0	0	0
Informational	4	0	0	1	0	1	2
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ILP	lending/interfaces/ILendingPool.sol	3d717bb98ad8d0fe858ccc4f1f55f982a52483d321a7e283a2d3cc0b828f2015
ILI	lending/interfaces/ILendingPoolInfo.sol	e19f6de53226a9062d04ae0b23dfd1c853c72f7a176ceff2cc427067260765a2
IPC	lending/interfaces/IPoolConfiguration.sol	18a1444494a078e796159270eea3d9c169184e7909cc0b4bd9b21de05abe6fd7
IPO	lending/interfaces/IPriceOracle.sol	f090568f812c2091c65143a4bc27a53edfceef57a54e6e8d759daddb6d876a86
IRB	lending/interfaces/IReceiver.sol	5cd7925b3a77734ebdebb66a69934ccf11757be18cd426fbfc6863df4ea303ce
ISR	lending/interfaces/IStdReference.sol	c93824faf39e8b76e35eb245d1e43b364d34d80a3d303fa4b9bcd59a050b785d
MAT	lending/libraries/Math.sol	8a064985cdb9e87acd43f78e4fdfba210b8a6c21b76dbab6f5f30bcf8e13516f
OBI	lending/libraries/Obi.sol	b164e0bbed478b169b62bdf05c94b53a0aeb7a9c7eeeebbc4784428c7943ef7b
WMB	lending/libraries/WadMath.sol	7a54725dd1e92f3e166a7460ecf5b9f154227f47c6c1397f5714ee0869b0febe
LPB	lending/LendingPool.sol	59834a0b2e954825edf2a51f892b4718032d582a938cdf9af2afe1a0aead1fa1
LPI	lending/LendingPoolInfo.sol	bc5475c956c4209608b314d8a9d4490e172c9d0b808eea95cf435fbe04011478
MTB	lending/MaToken.sol	3c2ad99997ce35fd706d5efa4fbe1bcf6e84fdabc5f06d8454a52b6aa2d40d67
MTD	lending/MaTokenDeployer.sol	a33536500696f4a3fa973ad4742c447a9bb95d27ff37b9365fed9078c1c492e2
PCB	lending/PoolConfiguration.sol	850ed914c47262cce6e79b4484446d7b8c9ec7546b376c122eed5ca347efeed5

Findings

Critical	**0** (0.00%)
Major	**5** (50.00%)
Medium	**0** (0.00%)
Minor	**1** (10.00%)
Informational	**4** (40.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
LEN-01	Missing Emit Events	Coding Style	Informational	Partially Resolved
LPB-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
LPB-02	Logical issue of the function `liquidate()`	Logical Issue	Minor	Acknowledged
LPB-03	Comparison to Boolean Constant	Coding Style	Informational	Resolved
LPI-01	Logical issue of the function `distributeMara()`	Logical Issue	Major	Resolved
LPI-02	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
LPI-03	Variables That Could Be Declared as Immutable	Gas Optimization	Informational	Resolved
MTB-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
MTB-02	Logical issue of the function `setTokenPerBlock()`	Logical Issue	Informational	Acknowledged
PCB-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged

LEN-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	lending/MaToken.sol: 85~87, 97~99 lending/LendingPool.sol: 274~276, 374~378, 1153~1155, 1183~1185 lending/LendingPoolInfo.sol: 65~67 lending/PoolConfiguration.sol: 60~76, 78~80, 82~84	Partially Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team heeded our advice and partially resolved this issue in commit `8db41d42ae2020cbb928d2871f84f38a01e2c641`.

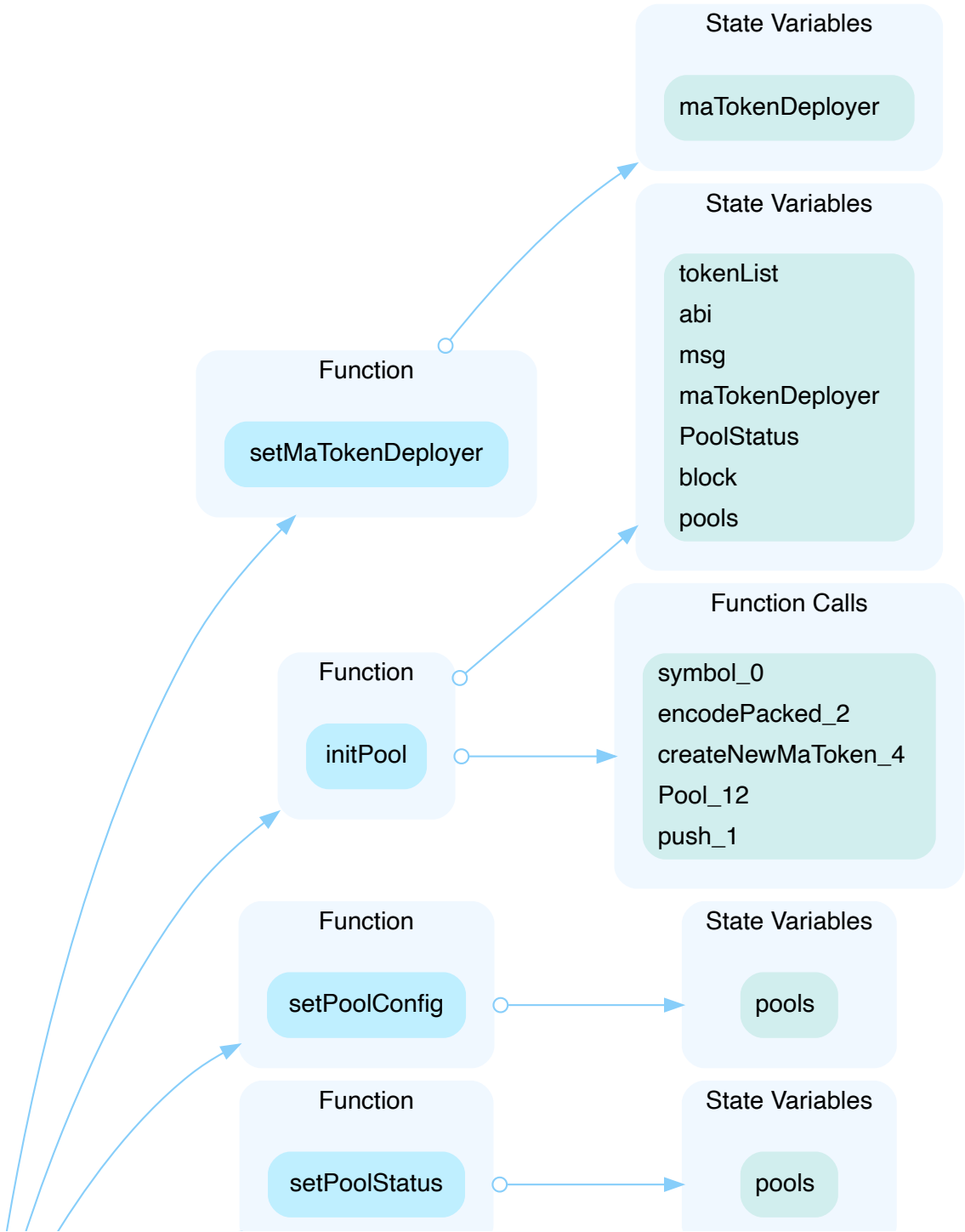
LPB-01 | Centralization Related Risks

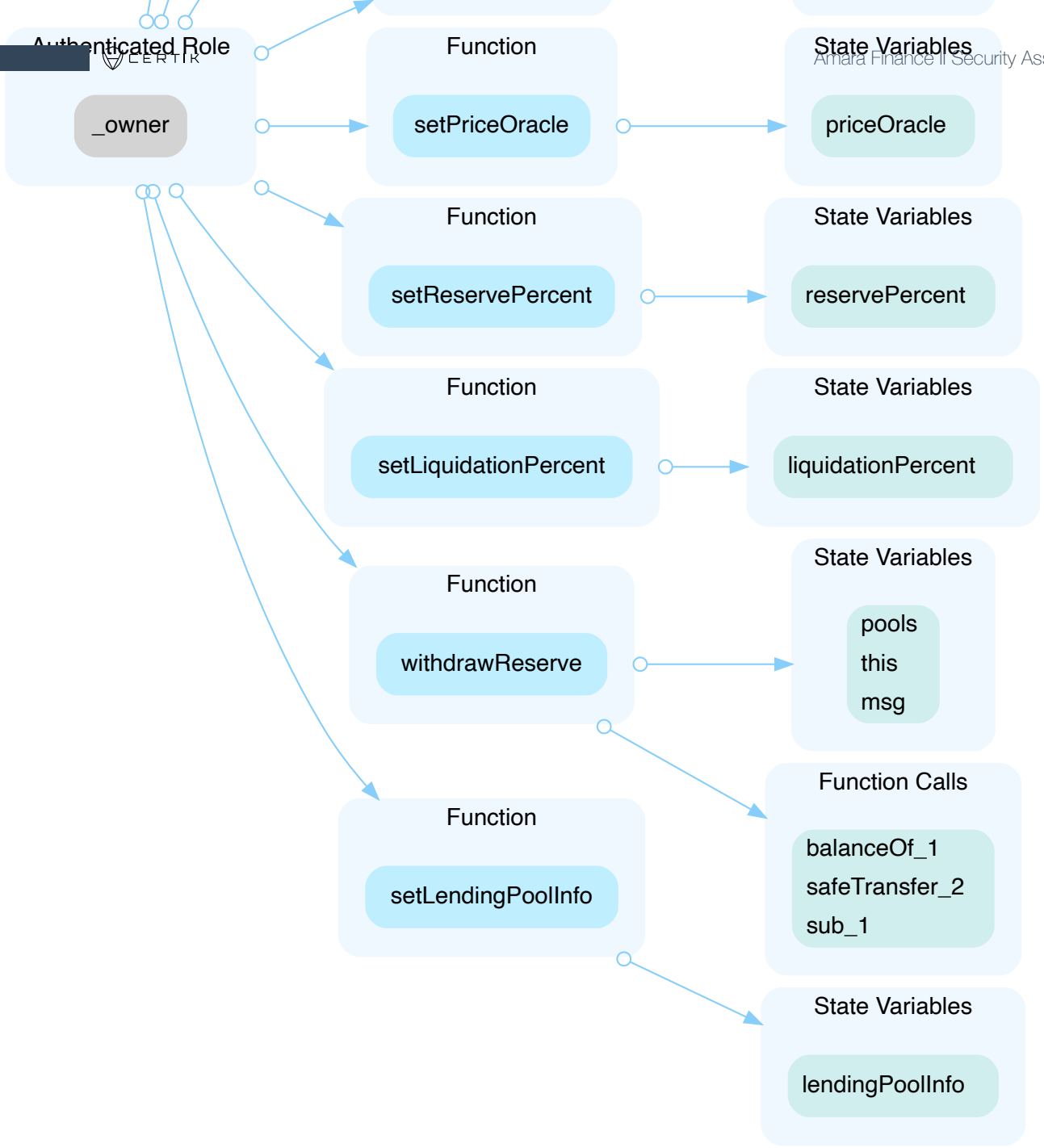
Category	Severity	Location	Status
Centralization / Privilege	Major	lending/LendingPool.sol: 274~276, 325~349, 359~367, 374~378, 398~401, 1143~1147, 1153~1155, 1162~1176, 1183~1185	Acknowledged

Description

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

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and [fixme, describe what hacker can do and the impact].





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

The team heeded our advice and they will transfer the ownership to the multi-signature wallet in their own timeframe.

LPB-02 | Logical Issue Of The Function `liquidate()`

Category	Severity	Location	Status
Logical Issue	Minor	lending/LendingPool.sol: 991	Acknowledged

Description

According to the following codes, the amount of the user to be liquidated is max up to half of the user's `borrowShares` in a pool for each call of the function `liquidate()`. The amount of collateral that the liquidator will receive as a reward will bonus an additional 5% amount according to the default parameter `liquidationBonus`.

```
1067      // 5. calculate liquidate amount and shares
1068      uint256 maxPurchaseShares = userTokenData.borrowShares.wadMul(CLOSE_FACTOR);
1069      uint256 liquidateShares = _liquidateShares;
1070      if (liquidateShares > maxPurchaseShares) {
1071        liquidateShares = maxPurchaseShares;
1072      }
1073      uint256 liquidateAmount = calculateRoundUpBorrowAmount(_token,
liquidateShares);
1074
1075      // 6. calculate collateral amount and shares
1076      uint256 collateralAmount = calculateCollateralAmount(_token, liquidateAmount,
_collateral);
1077      uint256 collateralShares = calculateRoundUpLiquidityShareAmount(_collateral,
collateralAmount);
```

As the values of the user's `borrowShares` in different pools may vary widely, the maximum amount that can be liquidated in a single call of the function `liquidate()` should be reasonably calculated.

Recommendation

We recommend stating for the logic of maximum amount that can be liquidated in a single call of the function `liquidate()`.

Alleviation

The team acknowledged this issue and they stated:

"This function is used to liquidate the user's debt until the user's account is healthy. Their codes reference the alpha finance project and the configuration is 50%. This is by design."

LPB-03 | Comparison To Boolean Constant

Category	Severity	Location	Status
Coding Style	Informational	lending/LendingPool.sol: 784	Resolved

Description

Boolean constants can be used directly and do not need to be compared to true or false.

File: contracts/lending/LendingPool.sol (Line 784, Function `LendingPool.borrow`)

```
require(pool.ableBorrow == true, "pool disable borrow, can't borrow this pool");
```

Recommendation

We recommend removing the equality to the boolean constant.

Alleviation

The team heeded our advice and resolved this issue in commit `a8a55ce8ede9b5876bbe75385d51644f853f21df`.

LPI-01 | Logical Issue Of The Function `distributeMara()`

Category	Severity	Location	Status
Logical Issue	Major	lending/LendingPoolInfo.sol: 72	Resolved

Description

According to the following codes, the function `distributeMara()` will set the variable `lastRewardBlock` to be zero when `block.number < startBlock`.

```
74      if (block.number < startBlock) {
75        lastRewardBlock = 0;
76        return;
77      }
```

When this condition is triggered, the call of the function `getMaraReleaseAmount()` will calculate the rewards to be `(toBlock - 0) * tokensPerBlock`. As a result, the reward calculation results in an additional calculation of `(startBlock - 0) * tokensPerBlock`.

Recommendation

We recommend setting the variable `lastRewardBlock` to be `startBlock` or returning directly.

Alleviation

The team heeded our advice and resolved this issue in commit `97c4ad2a3e6d6dcf8adbebd454638f9ff3599811`.

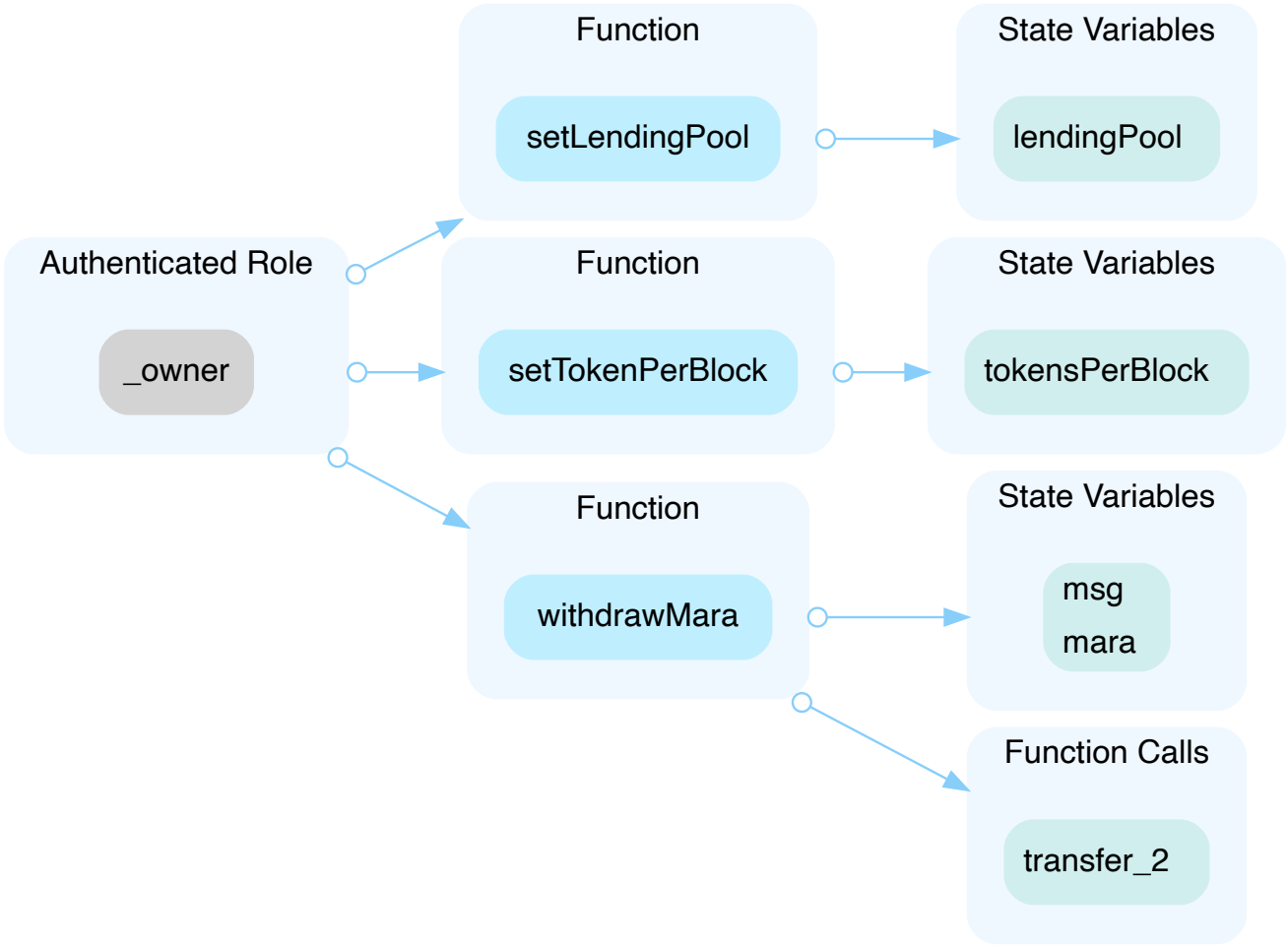
LPI-02 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	lending/LendingPoolInfo.sol: 60~63, 65~67, 90~93	Acknowledged

Description

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

Any compromise to the `_owner` 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

The team heeded our advice and they will transfer the ownership to the multi-signature wallet in their own timeframe.

LPI-03 | Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	lending/LendingPoolInfo.sol: 40	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. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The team heeded our advice and resolved this issue in commit `6a96a9e821d98f7fea7d801ce0afb5d3f3207d31`.

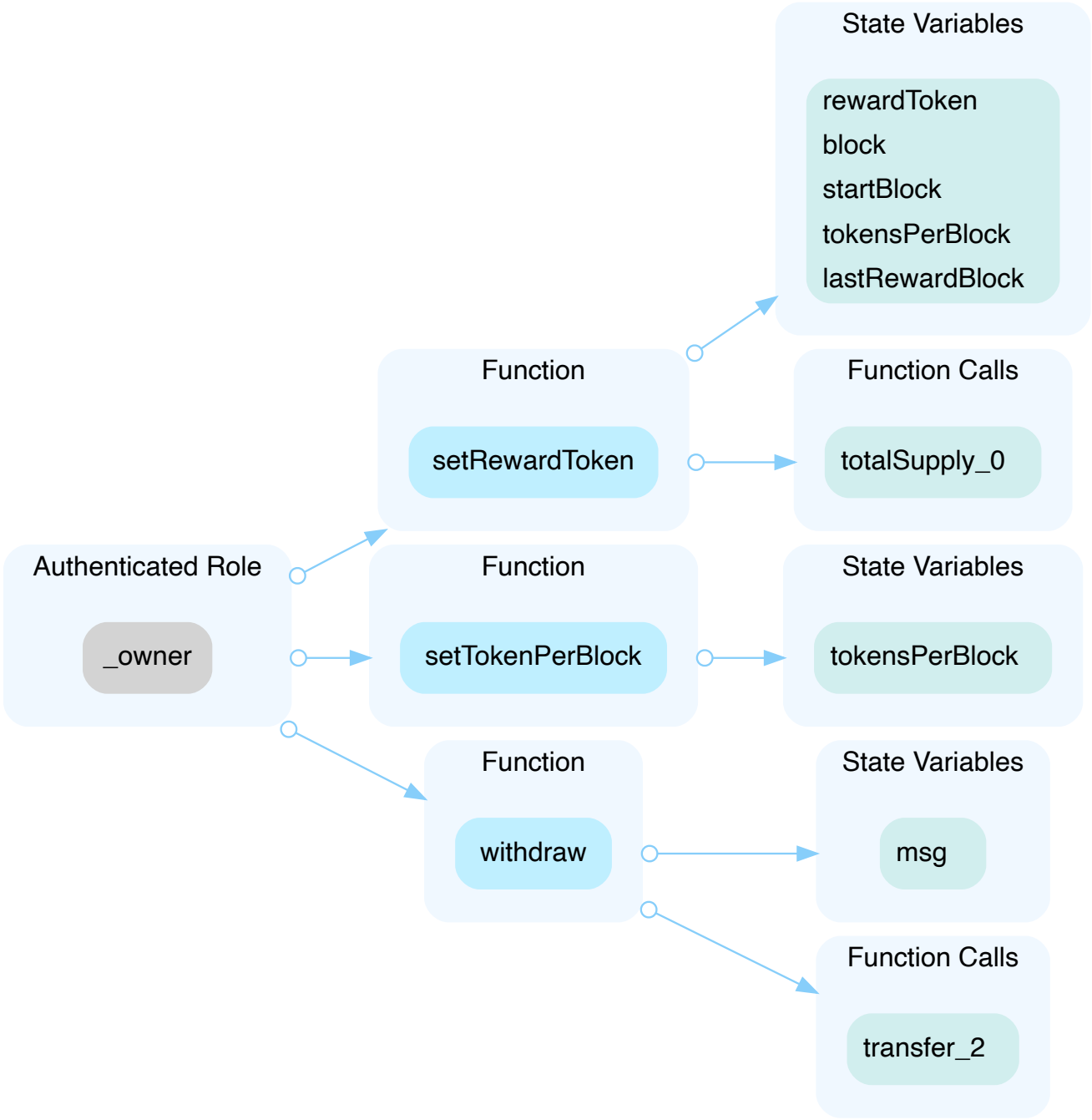
MTB-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	lending/MaToken.sol: 73~83, 85~87, 97~99	Acknowledged

Description

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

Any compromise to the `_owner` 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

The team heeded our advice and they will transfer the ownership to the multi-signature wallet in their own timeframe.

MTB-02 | Logical Issue Of The Function `setTokenPerBlock()`

Category	Severity	Location	Status
Logical Issue	Informational	lending/MaToken.sol: 85	Acknowledged

Description

The function `setTokenPerBlock()` is used to set the value of the variable `tokensPerBlock`, which is used to calculate the rewards. The rewards in the MaToken are not updated before the call of the function `setTokenPerBlock()`.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team acknowledged this issue and they stated:

"They will open the farming after setting the variable `tokensPerBlock`. The reward will be updated first if the variable `tokensPerBlock` will be modified."

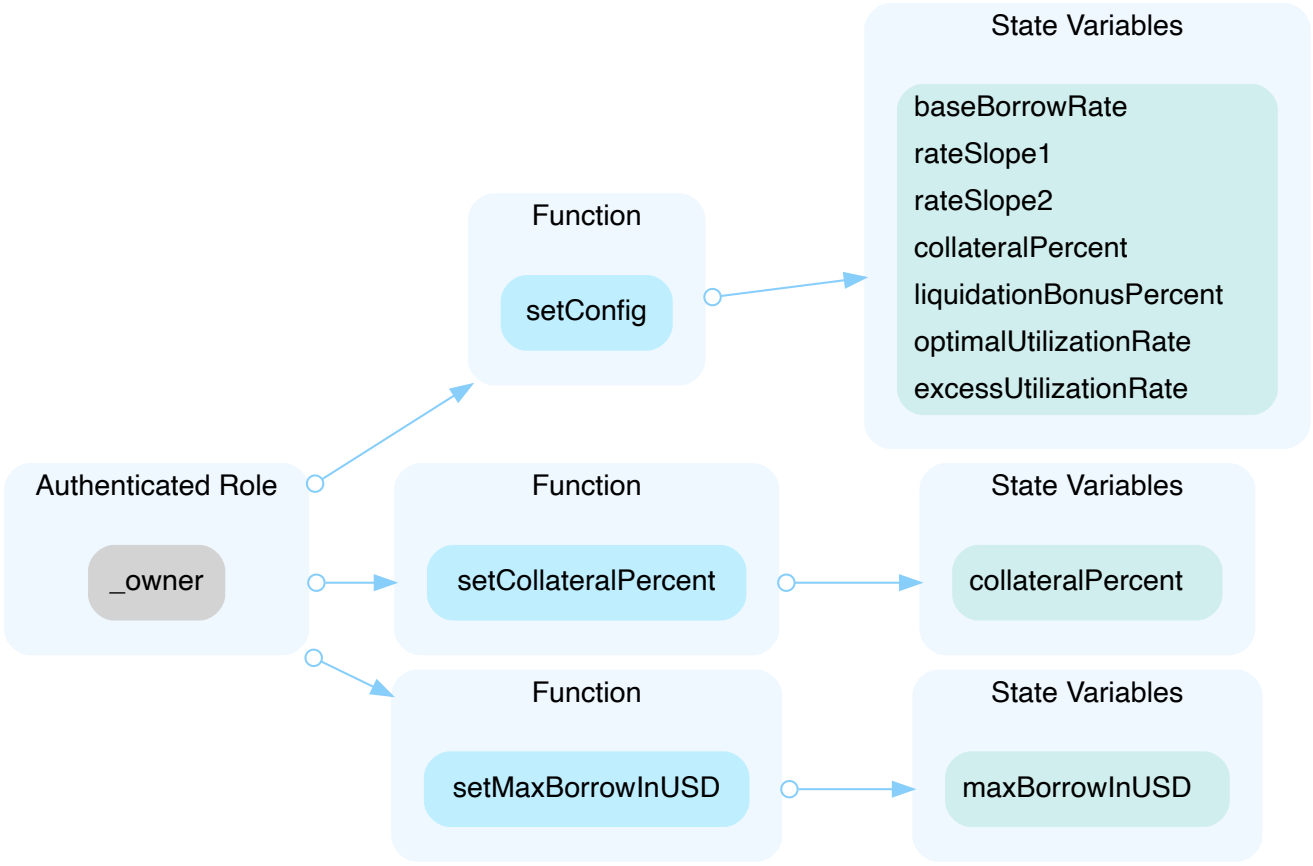
PCB-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	lending/PoolConfiguration.sol: 60~76, 78~80, 82~84	Acknowledged

Description

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

Any compromise to the `_owner` 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

The team heeded our advice and they will transfer the ownership to the multi-signature wallet in their own timeframe.

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.

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.