CERTIK

Security Assessment

Metanept

Jun 21st, 2022

Table of Contents

Summary

This report has been prepared for Metanept to discover issues and vulnerabilities in the source code of the Metanept 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	Metanept
Platform	Ethereum
Language	Solidity
Codebase	NFT721A 0x44646d197b8c0df61bff6e32858d3236f5dafbde NEPT 0x54d880b1c862e893d65accda8cf89e9224db1e4f

Audit Summary

Delivery Date	Jun 21, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

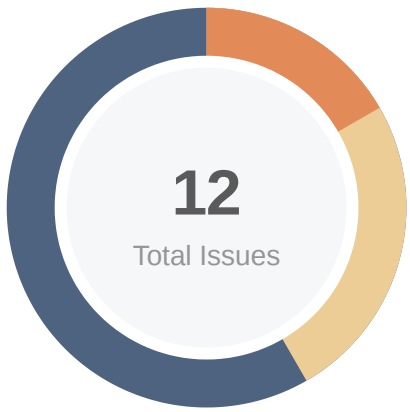
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	2	0	0	2	0	0	0
Medium	0	0	0	0	0	0	0
Minor	3	0	0	3	0	0	0
Optimization	2	0	0	2	0	0	0
Informational	7	0	0	7	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
NEP	NEPT.sol	9783268b5b6a2084055284226a2ead9add8217b68ec1db8810252c341744ba09
NFT	NFT721A_flat.sol	97a00dd763b6c1e8a6958a16065949c5f9741442fee01f544247414af8eb5765

Findings



Critical	**0** (0.00%)
Major	**2** (16.67%)
Medium	**0** (0.00%)
Minor	**3** (25.00%)
Informational	**7** (58.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
NEP-01	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
NEP-02	`constructor` Function Has Visibility `public`	Language Specific	Informational	Acknowledged
NEP-03	Inappropriate Symbol For A Token	Inconsistency	Informational	Acknowledged
NFT-01	Centralization Risks In NFT721A_flat.sol	**Centralization / Privilege**	**Major**	Acknowledged
NFT-02	No Upper Limit For Important Parameters	Logical Issue	Minor	Acknowledged
NFT-03	Usage Of `transfer()` For Sending Ether	Volatile Code	Minor	Acknowledged
NFT-04	Potentially Incorrect Validation Of `data`	Logical Issue	Minor	Acknowledged
NFT-07	Declaration Naming Convention	Coding Style	Informational	Acknowledged
NFT-08	Missing Emit Events	Language Specific	Informational	Acknowledged
NFT-09	Shadowing Local Variable	Coding Style	Informational	Acknowledged
NFT-10	Usage Of Hardhat's Console	Coding Style	Informational	Acknowledged
ROP-01	Unlocked Compiler Version	Language Specific	Informational	Acknowledged

NEP-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	NEPT.sol (NEPT): 538	Acknowledged

Description

All of the NEPT tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute NEPT tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

NEP-02 | `constructor` Function Has Visibility `public`

Category	Severity	Location	Status
Language Specific	Informational	NEPT.sol (NEPT): 537	Acknowledged

Description

Since Solidity v0.7.0 visibility (public / internal) is not needed for constructors anymore.

Recommendation

We recommend to remove visibility for `constructor` function.

NEP-03 | Inappropriate Symbol For A Token

Category	Severity	Location	Status
Inconsistency	Informational	NEPT.sol (NEPT): 537	Acknowledged

Description

The symbol `ERC-20` does not reflect the identity of this token and may mislead users.

Recommendation

We recommend changing the token symbol to something more appropriate for this project.

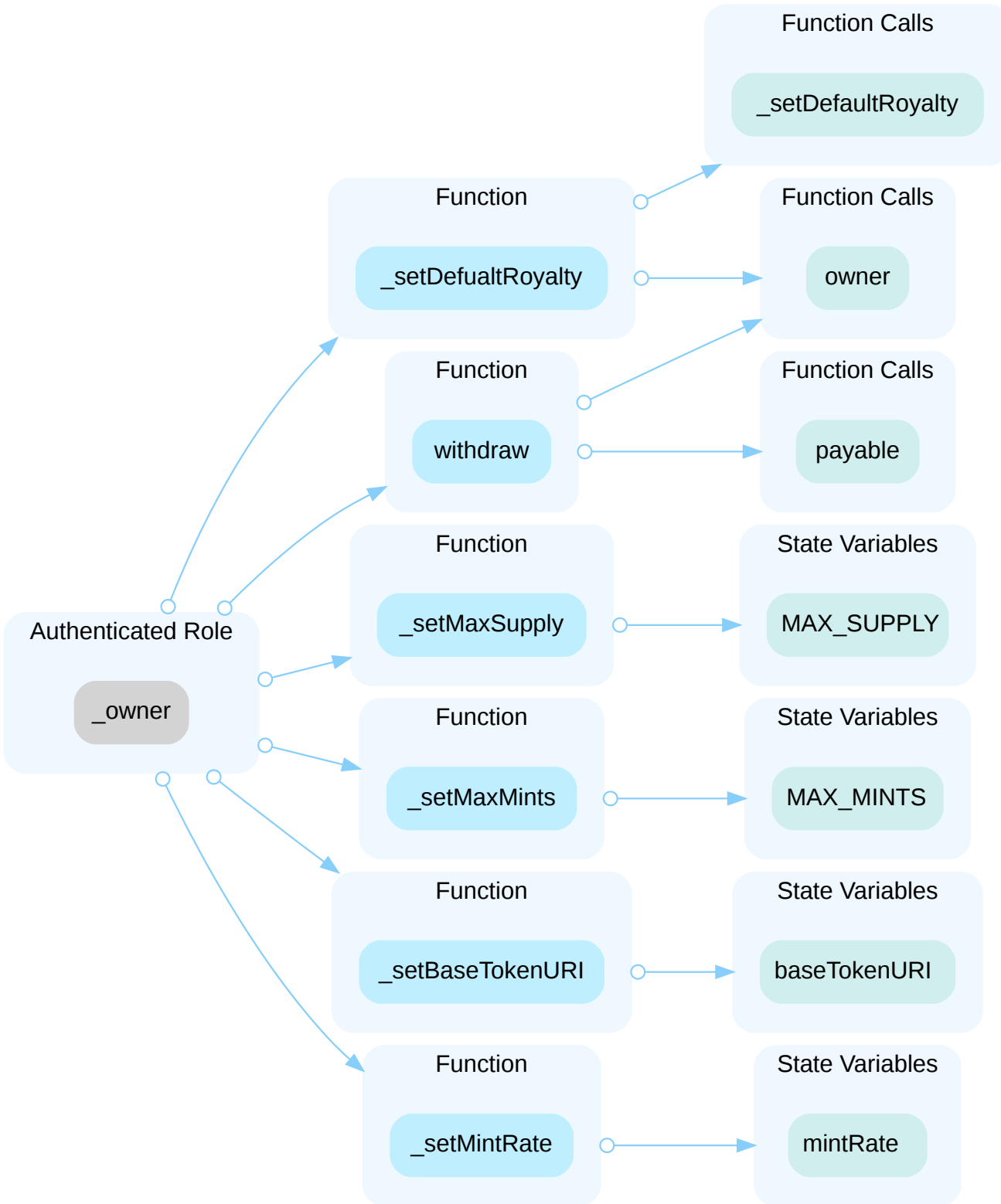
NFT-01 | Centralization Risks In NFT721A_flat.sol

Category	Severity	Location	Status
Centralization / Privilege	**Major**	NFT721A_flat.sol (NFT721A): 2997, 3036, 3058, 3062, 3066, 3070	Acknowledged

Description

In the contract `NFT721A` 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 to withdraw all funds and set the following variables:

- `baseTokenURI`;
- `mintRate`;
- `MAX_MINTS`;
- `MAX_SUPPLY`;
- `_royaltyFee`.



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.

NFT-02 | No Upper Limit For Important Parameters

Category	Severity	Location	Status
Logical Issue	Minor	NFT721A_flat.sol (NFT721A): 3062, 3066, 3070	Acknowledged

Description

The parameters below are sensitive to users' investment decisions, but are not set to a constant or have no upper limit:

- `MAX_MINTS`;
- `MAX_SUPPLY`;
- Default Royalty.

Recommendation

For more transparency, we recommend either setting these parameters as `constant` or `immutable`, or setting a reasonable upper limits for these parameters in setter functions based on the project design or description in WhitePaper.

NFT-03 | Usage Of `transfer()` For Sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	NFT721A_flat.sol (NFT721A): 2997~2999	Acknowledged

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

NFT-04 | Potentially Incorrect Validation Of `data`

Category	Severity	Location	Status
Logical Issue	Minor	NFT721A_flat.sol (NFT721A): 2993	Acknowledged

Description

The code checks that `data.length` is not equal to `1` before calling `_setTokenTURI(tokenMinted, data)`. Perhaps the check is not correct or additional conditions are missing.

The `_setTokenURI()` function doesn't check if there are enough `url` for `tokenMinted`. If there are too few or too many url's, either the token won't get a url, or the extra ones will be missing.

Recommendation

We recommend to recheck this part of the code and if the check is valid describe its meaning in the comment to improve the code readability. We also recommend adding an input data validation for `data`.

NFT-07 | Declaration Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	NFT721A_flat.sol (NFT721A): 2969, 2970, 3036, 3040, 3058, 3062, 3066, 3070, 3074	Acknowledged

Description

One or more declarations do not conform to the Solidity style guide with regards to its naming convention.

Particularly:

- The names of `public` or `external` functions begin with "_".
- `UPPER_CASE` used for non- `constant` variables.

Recommendation

We recommend adjusting those variable and function names to properly conform to Solidity's naming convention.

NFT-08 | Missing Emit Events

Category	Severity	Location	Status
Language Specific	Informational	NFT721A_flat.sol (NFT721A): 3059, 3063, 3067	Acknowledged

Description

No events are emitted during state changes to pass the changes out of chain.

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

NFT-09 | Shadowing Local Variable

Category	Severity	Location	Status
Coding Style	Informational	NFT721A_flat.sol (NFT721A): 2978, 2978, 3021	Acknowledged

Description

- Local variable `_name` in `NFT721A.constructor` shadows the variable `_name` in `ERC721A`.
- Local variable `_symbol` in `NFT721A.constructor` shadows the variable `_symbol` in `ERC721A`.
- Local variable `owner` in `NFT721A.getNftData` shadows the function `owner` in `Ownable`.

Recommendation

We recommend removing or renaming the local variable that shadows another definition.

NFT-10 | Usage Of Hardhat's Console

Category	Severity	Location	Status
Coding Style	Informational	NFT721A_flat.sol (NFT721A): 1420	Acknowledged

Description

The contract uses the console contract from Hardhat, which is meant to be used for testing purposes.

Recommendation

It is recommended to remove the import of Hardhat's contract for better code readability and simplicity.

ROP-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	NFT721A_flat.sol (NFT721A): 2958; NEPT.sol (NEPT): 532	Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an 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 advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.4` the contract should contain the following line:

```
pragma solidity 0.8.4;
```

Optimizations

ID	Title	Category	Severity	Status
NFT-05	Function Should Be Declared `external`	Gas Optimization	Optimization	Acknowledged
NFT-06	User-Defined Getters	Gas Optimization	Optimization	Acknowledged

NFT-05 | Function Should Be Declared `external`

Category	Severity	Location	Status
Gas Optimization	Optimization	NFT721A_flat.sol (NFT721A): 3026, 3058, 3062, 3066, 3070, 3074	Acknowledged

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

NFT-06 | User-Defined Getters

Category	Severity	Location	Status
Gas Optimization	Optimization	NFT721A_flat.sol (NFT721A): 3005~3007, 3009~3011	Acknowledged

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We advise that the linked variables are instead declared as `public` as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

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.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.