CERTIK

Security Assessment

MetaElfLand

May 15th, 2022

Table of Contents

Summary

This report has been prepared for MetaElfLand to discover issues and vulnerabilities in the source code of the MetaElfLand 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	MetaElfLand
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/MetaElfland/SolidityCore
Commit	7bfcd488a7a173fc6053d37e8d918a554f9d0770

Audit Summary

Delivery Date	May 15, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

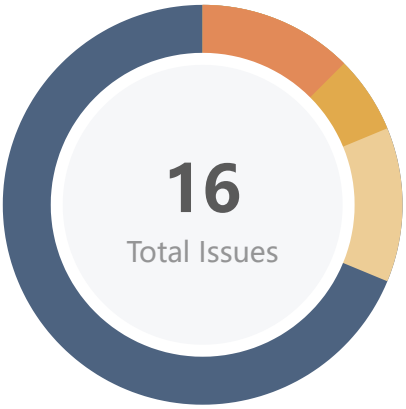
Vulnerability Summary

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	1	0	0	0	0	0	1
Minor	2	0	0	0	0	0	2
Informational	11	0	0	4	0	1	6
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
TSS	Token.sol	ac66819bc95d3a4ae71f9b8c6e0058e11f6f278719fe44bc8206b4047e31f79d
NFT	NFT.sol	fabb3038742c73bf2b064c2383f5f9fec8d41f546c1943f04f6c3befb9d14763
SSS	Store.sol	41de53ac6804161322eb9e8d4b7e09083383bb4f98f4385eff9f1d3967905c03

Findings



Critical	**0** (0.00%)
Major	**2** (12.50%)
Medium	**1** (6.25%)
Minor	**2** (12.50%)
Informational	**11** (68.75%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
NFT-01	Secondary Authorization	Logical Issue	Informational	Acknowledged
NFT-02	Unsound Value For `idToIndex[_tokenId]`	Logical Issue	Informational	Acknowledged
NFT-03	Dirty Data	Logical Issue	Informational	Acknowledged
SSC-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
SSC-02	Improper Usage Of `public` And `external` Type	Gas Optimization	Informational	Resolved
SSC-03	Missing Emit Events	Coding Style	Informational	Resolved
SSC-04	Unlocked Compiler Version	Language Specific	Informational	Resolved
SSS-01	No Upper Limit For `_fee`	Logical Issue	Medium	Resolved
SSS-02	Missing Input Validation	Volatile Code	Minor	Resolved
SSS-03	Potential Reentrancy Attack	Logical Issue	Minor	Resolved
SSS-04	Variables Never Used Can Be Removed	Gas Optimization	Informational	Resolved
SSS-05	Missing Error Messages	Coding Style	Informational	Partially Resolved
SSS-06	Variables That Could Be Declared As `constant`	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
TSS-01	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
TSS-02	Too Many Digits	Coding Style	Informational	Acknowledged
TSS-03	Dead Code	Coding Style	Informational	Resolved

NFT-01 | Secondary Authorization

Category	Severity	Location	Status
Logical Issue	Informational	NFT.sol: 588	Acknowledged

Description

Currently, both the `owner` of the `tokenId` and the owner's operator can set authorization, but usually, only the `owner` of `tokenId` can set authorization.

Recommendation

We advise the client to provide more details about this.

Alleviation

[Team]: In order to facilitate users to use batch operations, this authorized operator has been added, but we have made restrictions, and only the card can be authorized.

NFT-02 | Unsound Value For `idToIndex[_tokenId]`

Category	Severity	Location	Status
Logical Issue	Informational	NFT.sol: 884~900	Acknowledged

Description

The start index is 0 when the first NFT is mint, while the index will be set to 0 when burned and the default index value is also zero.

Recommendation

We advise setting the `idToIndex[_tokenId]` as `tokens.length` instead of `tokens.length-1`.

```
887         idToIndex[_tokenId] = tokens.length;
```

Alleviation

No alleviation.

NFT-03 | Dirty Data

Category	Severity	Location	Status
Logical Issue	Informational	NFT.sol: 905	Acknowledged

Description

When removing NFT, missing remove `idToOwnerIndex[_tokenId]`.

Recommendation

We advise removing `idToOwnerIndex[_tokenId]`.

```
905         uint256 tokenToRemoveIndex = idToOwnerIndex[_tokenId];
906         uint256 lastTokenIndex = ownerToIds[_from].length - 1;
907 
908         if (lastTokenIndex != tokenToRemoveIndex){
909             uint256 lastToken = ownerToIds[_from][lastTokenIndex];
910             ownerToIds[_from][tokenToRemoveIndex] = lastToken;
911             idToOwnerIndex[lastToken] = tokenToRemoveIndex;
912         }
913         ownerToIds[_from].pop();
914         delete idToOwnerIndex[_tokenId];
```

Alleviation

No alleviation.

SSC-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	NFT.sol; Store.sol; Token.sol	ⓘ Acknowledged

Description

In the contract `MetaElfLand`, the role `owner` has authority over the following functions:

- function mint()
- function renounceOwnership()
- function transferOwnership()

In the contract `zNFTCONTRACT`, the role `manager` has authority over the following functions:

- function mint() will mint NFT to anyone.
- function burn(() will burn anyone's NFT.
- function setStarall1()
- function setStarall2()
- function setStarall3()
- function setStarall4()
- function setStarall5()
- function setStarall6()

In the contract `zNFTCONTRACT`, the role `owner` has authority over the following functions:

- function addManager()
- function delManager()
- function transferOwnership()

In the contract `Consignment`, the role `governance` has authority over the following functions:

- function setGovernance()
- function setFee()
- function withdraw()
- function setNFTAddr()

Any compromise to these accounts may allow a 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., multi-signature wallets.

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

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Alleviation

[Team]: We will delete MINT and give up the administrator authority. After NFT gamification, the NFT attribute is involved in the chain, so there is an administrator authority to manage the attribute. For the convenience of management, a secondary administrator authority is added. The mall contract administrator sets the transaction fee management We will transfer the permissions to the multi-signature wallet.

SSC-02 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	NFT.sol: 397, 401, 404, 516, 611, 954, 957; Token.sol: 316, 325, 465, 484, 497, 505; Store.sol: 35, 312, 321, 704, 708, 736, 756	Resolved

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSC-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	NFT.sol: 397, 401, 963, 980, 995, 1010, 1025, 1040; Token.sol: 325, 497; Store.sol: 704, 756	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 development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSC-04 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	NFT.sol; Store.sol	Resolved

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 `v0.5.8` the contract should contain the following line:

```
pragma solidity 0.5.8;
```

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-01 | No Upper Limit For `_fee`

Category	Severity	Location	Status
Logical Issue	Medium	Store.sol: 704	Resolved

Description

In the current implementation, there is no upper limit for the fee rate. Misuse of these fee setting functions could damage the whole tokenomics. For example, the owner can set the fee rate to more than 100% to cause all transactions to revert.

Recommendation

We recommend setting a reasonable upper limit of `_fee` such as 1000. (10%)

Alleviation

The development team has set the upper limit of `_fee` as `10000`(100%) in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	Store.sol: 624, 756	Resolved

Description

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

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

```
constructor(address _nftaddr) public {
    require(_nftaddr != address(0), "_nftaddr is the zero address");
    nftAddr = _nftaddr;
}

function setNFTAddr(address newnftaddr) public onlyGovernance{
    require(newnftaddr != address(0), "newnftaddr is the zero address");
    nftAddr = newnftaddr;
}
```

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-03 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	Minor	Store.sol: 628, 649, 673	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

- function sell(uint _tokenid,uint _price)
- function cancelSell(uint _tokenid)
- function buy(uint _tokenid)

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-04 | Variables Never Used Can Be Removed

Category	Severity	Location	Status
Gas Optimization	Informational	Store.sol: 610~614	Resolved

Description

These variables `_nftTypeInfo/_nftNameInfo/_nftURLInfo/_nftBrandInfo/_nftNumberingInfo` are never used.

Recommendation

We advise the client to remove these unused variables.

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-05 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	Store.sol: 629, 630, 650, 651	Partially Resolved

Description

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

Recommendation

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

Alleviation

The development team partially resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

SSS-06 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	Store.sol: 617, 621	Resolved

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

TSS-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Token.sol: 356	Acknowledged

Description

All of the `MELT` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute all 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.

Alleviation

[Team]: Tokens will be directly transferred to the corresponding multi-signature permission wallet and mining contract.

TSS-02 | Too Many Digits

Category	Severity	Location	Status
Coding Style	Informational	Token.sol: 355	Acknowledged

Description

Literals with many digits are difficult to read and review.

File: Solidity/Token.sol (Line 355, Function `MetaElfLand.constructor`)

```
    _totalSupply = 1000000000*1e8;
```

Recommendation

We advise the client to use the scientific notation to improve readability.

Alleviation

No alleviation.

TSS-03 | Dead Code

Category	Severity	Location	Status
Coding Style	Informational	Token.sol: 596~599	Resolved

Description

The internal function `_burnFrom` is not used.

Recommendation

We recommend removing the unused function.

Alleviation

The development team resolved this issue in commit `609452f2784c1924009b6d532cff7cee48159398`.

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.

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.