CERTIK

Security Assessment

TrustFi Network

Jun 4th, 2021

Table of Contents

Summary

This report has been prepared for TrustFi Network smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Formal Verification 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 1 informational finding. 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	TrustFi Network
Description	TrustFi Network is a decentralized BaaS solution for DeFi market based on multichain environment and focusing on early crypto assets issuance, liquidity management, community activities and DAO governance to unlock the potential of DeFi.
Platform	Ethereum, BSC
Language	Solidity
Codebase	
Commits	

Audit Summary

Delivery Date	Jun 04, 2021
Audit Methodology	Formal Verification
Key Components	

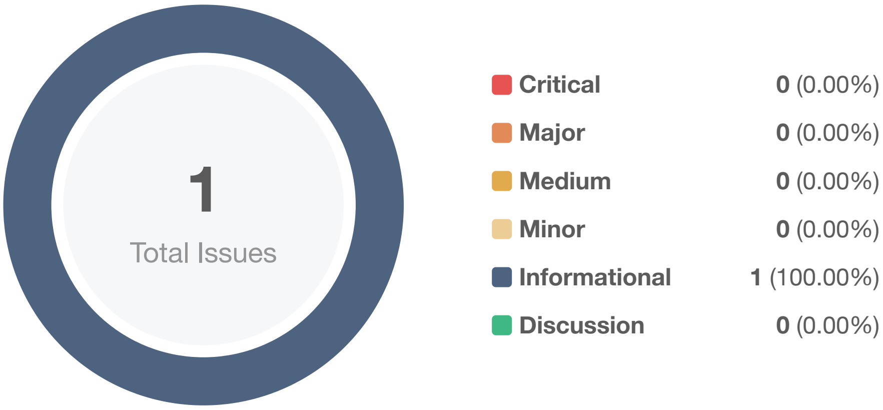
Vulnerability Summary

Total Issues	1
● Critical	0
● Major	0
● Medium	0
● Minor	0
● Informational	1
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
TTF	contracts/Token.sol	b31cfd18a88207c2c8653bb2acb5f17a8b2c148ed741abccc165fc64801e122f

Findings



ID	Title	Category	Severity	Status
TTF-01	Unlocked Compiler Version Declaration	Language Specific	● Informational	⊗ Declined

TTF-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	● Informational	contracts/Token.sol: 5, 31, 110, 271, 414, 721	⊗ Declined

Description

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 also 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

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones 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.

Formal Verification Requests

Request 1 | Context _msgSender

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Context	_msgSender

Contract Code

```
function _msgSender() internal view virtual returns (address payable) {
    return msg.sender;
}
```

CertiK Label

```
/*@CTK  "Context _msgSender"
  @post __return == msg.sender
*/
```

✓ The code meets the specification.

Request 2 | Context _msgData

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Context	_msgData

Contract Code

```
    function _msgData() internal view virtual returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
```

CertiK Label

```
    /*@CTK  "Context _msgData"
      @post __return == msg.data
    */
```

✓ The code meets the specification.

Request 3 | SafeMath add

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	add

Contract Code

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");

    return c;
}
```

CertiK Label

```
/*@CTK "SafeMath add"
 @post (a + b < a || a + b < b) == __reverted
 @post !__reverted -> __return == a + b
 @post !__reverted -> !__has_overflow
 @post !__reverted -> !__has_assertion_failure
 @post !(__has_buf_overflow)
*/
```

✓ The code meets the specification.

Request 4 | SafeMath sub

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	sub

Contract Code

```
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");
}
```

CertiK Label

```
/*@CTK "SafeMath sub"
 @post (a < b) == __reverted
 @post !__reverted -> __return == a - b
 @post !__reverted -> !__has_overflow
 @post !__reverted -> !__has_assertion_failure
 @post !(__has_buf_overflow)
*/
```

✓ The code meets the specification.

Request 5 | SafeMath sub_wth_message

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	sub

Contract Code

```
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
```

CertiK Label

```
     /*@CTK "SafeMath sub_wth_message"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
```

✓ The code meets the specification.

Request 6 | SafeMath mul

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	mul

Contract Code

```
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
    if (a == 0) {
        return 0;
    }

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;
}
```

CertiK Label

```
/*@CTK "SafeMath mul"
 @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
 @post !__reverted -> __return == a * b
 @post !__reverted == !__has_overflow
 @post !__reverted -> !__has_assertion_failure
 @post !(__has_buf_overflow)
*/
```

✓ The code meets the specification.

Request 7 | SafeMath div

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	div

Contract Code

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }
```

CertiK Label

```
     /*@CTK "SafeMath div"
      @post (b <= 0) == __reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
```

✓ The code meets the specification.

Request 8 | SafeMath div_with_message

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	div

Contract Code

```
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }
```

CertiK Label

```
     /*@CTK "SafeMath div_with_message"
      @post (b <= 0) == __reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
```

✓ The code meets the specification.

Request 9 | SafeMath mod

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	mod

Contract Code

```
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
}
```

CertiK Label

```
 /*@CTK "SafeMath mod"
  @post (b == 0) == __reverted
  @post !__reverted -> __return == a % b
  @post !__reverted -> !__has_overflow
  @post !__reverted -> !__has_assertion_failure
  @post !(__has_buf_overflow)
*/
```

✓ The code meets the specification.

Request 10 | SafeMath mod_with_message

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	SafeMath	mod

Contract Code

```
    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
```

CertiK Label

```
     /*@CTK "SafeMath mod_with_message"
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
```

✓ The code meets the specification.

Request 11 | ERC20 constructor

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	__constructor__ERC20

Contract Code

```
constructor (string memory name, string memory symbol) public {
    _name = name;
    _symbol = symbol;
    _decimals = 18;
}
```

CertiK Label

```
 /*@CTK "ERC20 constructor"
  @post __post._name == name
  @post __post._symbol == symbol
  @post __post._decimals == 18
*/
```

✓ The code meets the specification.

Request 12 | ERC20 name

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	name

Contract Code

```
function name() public view returns (string memory) {
    return _name;
}
```

CertiK Label

```
 /*@CTK "ERC20 name"
  @post __return == _name
*/
```

✓ The code meets the specification.

Request 13 | ERC20 symbol

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	symbol

Contract Code

```
    function symbol() public view returns (string memory) {
        return _symbol;
    }
```

CertiK Label

```
     /*@CTK "ERC20 symbol"
      @post __return == _symbol
    */
```

✓ The code meets the specification.

Request 14 | ERC20 decimals

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	decimals

Contract Code

```
function decimals() public view returns (uint8) {
    return _decimals;
}
```

CertiK Label

```
 /*@CTK "ERC20 decimals"
  @post __return == _decimals
*/
```

✓ The code meets the specification.

Request 15 | ERC20 totalSupply

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	totalSupply

Contract Code

```
function totalSupply() public view override returns (uint256) {
    return _totalSupply;
}
```

CertiK Label

```
 /*@CTK "ERC20 totalSupply"
  @post __return == _totalSupply
*/
```

✓ The code meets the specification.

Request 16 | ERC20 balanceOf

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	balanceOf

Contract Code

```
function balanceOf(address account) public view override returns (uint256) {
    return _balances[account];
}
```

CertiK Label

```
 /*@CTK "ERC20 balanceOf"
  @post __return == _balances[account]
*/
```

✓ The code meets the specification.

Request 17 | ERC20 transfer

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	transfer

Contract Code

```
    function transfer(address recipient, uint256 amount) public virtual override returns
(bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }
```

CertiK Label

```
     /*@CTK "ERC20 transfer"
     @tag assume_completion
      @pre msg.sender != address(0)
      @pre recipient != address(0)
      @post msg.sender != recipient -> __post._balances[recipient] ==
_balances[recipient] + amount
      @post msg.sender != recipient -> __post._balances[msg.sender] ==
_balances[msg.sender] - amount
      @post msg.sender == recipient -> __post._balances[msg.sender] ==
_balances[msg.sender]
     */
```

✓ The code meets the specification.

Request 18 | ERC20 allowance

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	allowance

Contract Code

```
    function allowance(address owner, address spender) public view virtual override
returns (uint256) {
        return _allowances[owner][spender];
    }
```

CertiK Label

```
     /*@CTK "ERC20 allowance"
      @post __return == _allowances[owner][spender]
    */
```

✓ The code meets the specification.

Request 19 | ERC20 apprrove

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	approve

Contract Code

```
    function approve(address spender, uint256 amount) public virtual override returns
(bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }
```

CertiK Label

```
     /*@CTK "ERC20 apprrove"
      @pre msg.sender != address(0)
      @pre spender != address(0)
      @post __post._allowances[msg.sender][spender] == amount
    */
```

✓ The code meets the specification.

Request 20 | ERC20 transferFrom

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	transferFrom

Contract Code

```
    function transferFrom(address sender, address recipient, uint256 amount) public
virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
"ERC20: transfer amount exceeds allowance"));
        return true;
    }
```

CertiK Label

```
     /*@CTK "ERC20 transferFrom"
      @tag assume_completion
      @pre sender != address(0)
      @pre recipient != address(0)
      @pre msg.sender != address(0)
      @post sender != recipient -> __post._balances[recipient] == _balances[recipient] +
amount
      @post sender != recipient -> __post._balances[sender] == _balances[sender] - amount
      @post sender == recipient -> __post._balances[sender] == _balances[sender]
      @post __post._allowances[sender][msg.sender] == _allowances[sender][msg.sender] -
amount
    */
```

✓ The code meets the specification.

Request 21 | ERC20 increaseAllowance

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	increaseAllowance

Contract Code

```
    function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()]
[spender].add(addedValue));
        return true;
    }
```

CertiK Label

```
     /*@CTK "ERC20 increaseAllowance"
      @tag assume_completion
      @post __post._allowances[msg.sender][spender] == _allowances[msg.sender][spender] +
addedValue
    */
```

✓ The code meets the specification.

Request 22 | ERC20 decreaseAllowance

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	decreaseAllowance

Contract Code

```
    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual
returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()]
[spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
        return true;
    }
```

CertiK Label

```
     /*@CTK "ERC20 decreaseAllowance"
      @tag assume_completion
      @post __post._allowances[msg.sender][spender] == _allowances[msg.sender][spender] -
subtractedValue
    */
```

✓ The code meets the specification.

Request 23 | ERC20 _transfer

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	_transfer

Contract Code

```
    function _transfer(address sender, address recipient, uint256 amount) internal
virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds
balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }
```

CertiK Label

```
     /*@CTK "ERC20 _transfer"
      @tag assume_completion
      @pre sender != address(0)
      @pre recipient != address(0)
      @post sender != recipient -> __post._balances[recipient] == _balances[recipient] +
amount
      @post sender != recipient -> __post._balances[sender] == _balances[sender] - amount
      @post sender == recipient -> __post._balances[sender] == _balances[sender]
    */
```

✓ The code meets the specification.

Request 24 | ERC20 _mint

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	_mint

Contract Code

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);
    emit Transfer(address(0), account, amount);
}
```

CertiK Label

```
 /*@CTK "ERC20 _mint"
  @tag assume_completion
  @pre account != address(0)
  @post __post._totalSupply == _totalSupply + amount
  @post __post._balances[account] == _balances[account] + amount
*/
```

✓ The code meets the specification.

Request 25 | ERC20 _burn

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	_burn

Contract Code

```
    function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");

        _beforeTokenTransfer(account, address(0), amount);

        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds
balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }
```

CertiK Label

```
     /*@CTK "ERC20 _burn"
      @tag assume_completion
      @pre account != address(0)
      @post __post._totalSupply == _totalSupply - amount
      @post __post._balances[account] == _balances[account] - amount
    */
```

✓ The code meets the specification.

Request 26 | ERC20 _approve

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	_approve

Contract Code

```
function _approve(address owner, address spender, uint256 amount) internal virtual {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

CertiK Label

```
/*@CTK "ERC20 _approve"
  @pre owner != address(0)
  @pre spender != address(0)
  @post __post._allowances[owner][spender] == amount
*/
```

✓ The code meets the specification.

Request 27 | ERC20 _setupDecimals

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	ERC20	_setupDecimals

Contract Code

```
function _setupDecimals(uint8 decimals_) internal {
    _decimals = decimals_;
}
```

CertiK Label

```
/*@CTK "ERC20 _setupDecimals"
@post __post._decimals == decimals_
*/
```

✓ The code meets the specification.

Request 28 | If method completes, integer overflow would not happen.

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Token	__constructor__Token

Contract Code

```
constructor() public  ERC20("TrustFi Network Token", "TFI") {
    _mint(0xE15CD3FB9315e72A319f80cf7442815c5f3618Ec, 10 ** 8 * 10 ** 18);
}
```

CertiK Label

```
//@CTK NO_OVERFLOW
```

✓ The code meets the specification.

Request 29 | Buffer overflow / array index out of bound would never happen.

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Token	__constructor__Token

Contract Code

```
constructor() public  ERC20("TrustFi Network Token", "TFI") {
    _mint(0xE15CD3FB9315e72A319f80cf7442815c5f3618Ec, 10 ** 8 * 10 ** 18);
}
```

CertiK Label

```
//@CTK NO_BUF_OVERFLOW
```

✓ The code meets the specification.

Request 30 | Method will not encounter an assertion failure.

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Token	__constructor__Token

Contract Code

```
constructor() public  ERC20("TrustFi Network Token", "TFI") {
    _mint(0xE15CD3FB9315e72A319f80cf7442815c5f3618Ec, 10 ** 8 * 10 ** 18);
}
```

CertiK Label

```
//@CTK NO_ASF
```

✓ The code meets the specification.

Request 31 | Toekn constructor

Status	Details	File	Contract	Method
✓ Success	The code meets the specification.	Token.sol	Token	__constructor__Token

Contract Code

```
    constructor() public  ERC20("TrustFi Network Token", "TFI") {
        _mint(0xE15CD3FB9315e72A319f80cf7442815c5f3618Ec, 10 ** 8 * 10 ** 18);
    }
```

CertiK Label

```
    /*@CTK "Toekn constructor"
      @tag assume_completion
      @pre _totalSupply == 0
      @post __post._name == "TrustFi Network Token"
      @post __post._symbol == "TFI"
      @post __post._decimals == 18
      @post __post._totalSupply == 100000000 * (10 ** 18)
      @post __post._balances[0xE15CD3FB9315e72A319f80cf7442815c5f3618Ec] == 100000000 *
(10 ** 18)
     */
```

✓ The code meets the specification.

Appendix

Finding Categories

Language Specific

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

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 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 without CertiK's prior written consent.

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.

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.

CERTIK