Security Assessment

Orange Cat Token

Jun 16th, 2021

Table of Contents

Summary

Overview

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

Formal Verification Requests

Appendix

Disclaimer

About

Summary

This report has been prepared for Orange Cat Token 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 0 finding 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 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	Orange Cat Token
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x7128e52ca302c5f5beeb801b6ad373fdebe3dc5e#code
Commit	

Audit Summary

Delivery Date	Jun 16, 2021
Audit Methodology	Formal Verification
Key Components	

Vulnerability Summary

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

Audit Scope

ID	file	SHA256 Checksum
BEP	contracts/BEP20Token.sol	78991f16e1cfbf80fc918d2622f891afe04055be5217416486e3464c0d6443aa

Certralization Roles

The Orange Cat Token smart contract introduces an authorization.

Owner:

- mint(): mint token to `owner`.

Formal Verification Requests

Request 1 | Context _msgSender

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	Context	_msgSender

Contract Code

```
function _msgSender() internal view 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.	BEP20Token.sol	Context	_msgData

Contract Code

```
  function _msgData() internal view 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.	BEP20Token.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.	BEP20Token.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.	BEP20Token.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.	BEP20Token.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.	BEP20Token.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.	BEP20Token.sol	SafeMath	div

Contract Code

```
  function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
    // Solidity only automatically asserts when dividing by 0
    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.	BEP20Token.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.	BEP20Token.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 | Ownable constructor

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	Ownable	__constructor__Ownable

Contract Code

```
constructor () internal {
  address msgSender = _msgSender();
  _owner = msgSender;
  emit OwnershipTransferred(address(0), msgSender);
}
```

CertiK Label

```
/*@CTK "Ownable constructor"
   @post __post._owner == msg.sender
 */
```

The code meets the specification.

Request 12 | Ownable owner

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	Ownable	owner

Contract Code

```
function owner() public view returns (address) {
  return _owner;
}
```

CertiK Label

```
/*@CTK "Ownable owner"
   @post __return == _owner
 */
```

The code meets the specification.

Request 13 | Ownable renounceOwnership

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	Ownable	renounceOwnership

Contract Code

```
function renounceOwnership() public onlyOwner {
  emit OwnershipTransferred(_owner, address(0));
  _owner = address(0);
}
```

CertiK Label

```
/*@CTK "Ownable renounceOwnership"
   @pre msg.sender == _owner
   @post __post._owner == address(0)
 */
```

The code meets the specification.

Request 14 | Ownable transferOwnership

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	Ownable	_transferOwnership

Contract Code

```
function _transferOwnership(address newOwner) internal {
  require(newOwner != address(0), "Ownable: new owner is the zero address");
  emit OwnershipTransferred(_owner, newOwner);
  _owner = newOwner;
}
```

CertiK Label

```
/*@CTK "Ownable transferOwnership"
 @pre msg.sender == __post._owner
 @pre newOwner != address(0)
 @post __post._owner == newOwner
 */
```

The code meets the specification.

Request 15 | If method completes, integer overflow would not happen.

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	__constructor__BEP20Token

Contract Code

```
constructor() public {
  _name = "Orange Cat Token";
  _symbol = "OCAT";
  _decimals = 18;
  _totalSupply = 100000000000000000000000000000000;
  _balances[msg.sender] = _totalSupply;

  emit Transfer(address(0), msg.sender, _totalSupply);
}
```

CertiK Label

```
//@CTK NO_OVERFLOW
```

The code meets the specification.

Request 16 | Buffer overflow / array index out of bound would never happen.

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	__constructor__BEP20Token

Contract Code

```
constructor() public {
  _name = "Orange Cat Token";
  _symbol = "OCAT";
  _decimals = 18;
  _totalSupply = 100000000000000000000000000000000;
  _balances[msg.sender] = _totalSupply;

  emit Transfer(address(0), msg.sender, _totalSupply);
}
```

CertiK Label

```
//@CTK NO_BUF_OVERFLOW
```

The code meets the specification.

Request 17 | Method will not encounter an assertion failure.

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	__constructor__BEP20Token

Contract Code

```
constructor() public {
  _name = "Orange Cat Token";
  _symbol = "OCAT";
  _decimals = 18;
  _totalSupply = 100000000000000000000000000000000;
  _balances[msg.sender] = _totalSupply;

  emit Transfer(address(0), msg.sender, _totalSupply);
}
```

CertiK Label

```
  //@CTK NO_ASF
```

The code meets the specification.

Request 18 | BEP20Token constructor

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	__constructor__BEP20Token

Contract Code

```
constructor() public {
  _name = "Orange Cat Token";
  _symbol = "OCAT";
  _decimals = 18;
  _totalSupply = 100000000000000000000000000000000;
  _balances[msg.sender] = _totalSupply;

  emit Transfer(address(0), msg.sender, _totalSupply);
}
```

CertiK Label

```
/*@CTK "BEP20Token constructor"
  @tag assume_completion
  @post __post._name == "Orange Cat Token"
  @post __post._symbol == "OCAT"
  @post __post._decimals == 18
  @post __post._totalSupply == 100000000000000000000000000000000
  @post __post._balances[msg.sender] == _totalSupply
 */
```

The code meets the specification.

Request 19 | BEP20Token getOwner

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	getOwner

Contract Code

```
function getOwner() external view returns (address) {
  return owner();
}
```

CertiK Label

```
/*@CTK "BEP20Token getOwner"
@post __return == _owner
*/
```

The code meets the specification.

Request 20 | BEP20Token decimals

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	decimals

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token decimals"
@post __return == _decimals
*/
```

The code meets the specification.

Request 21 | BEP20Token symbol

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	symbol

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token symbol"
@post __return == _symbol
*/
```

The code meets the specification.

Request 22 | BEP20Token name

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	name

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token name"
 @post __return == _name
 */
```

The code meets the specification.

Request 23 | BEP20Token totalSupply

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	totalSupply

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token totalSupply"
@post __return == _totalSupply
*/
```

The code meets the specification.

Request 24 | BEP20Token balanceOf

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	balanceOf

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token balanceOf"
@post __return == _balances[account]
*/
```

The code meets the specification.

Request 25 | BEP20Token transfer

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	transfer

Contract Code

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

CertiK Label

```
   /*@CTK "BEP20Token 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 26 | BEP20Token allowance

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	allowance

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token allowance"
@post __return == _allowances[owner][spender]
*/
```

The code meets the specification.

Request 27 | BEP20Token approve

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	approve

Contract Code

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

CertiK Label

```
/*@CTK "BEP20Token approve"
    @pre msg.sender != address(0)
    @pre spender != address(0)
    @post __post._allowances[msg.sender][spender] == amount
    @post __return == true
  */
```

The code meets the specification.

Request 28 | BEP20Token transferFrom

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	transferFrom

Contract Code

```
  function transferFrom(address sender, address recipient, uint256 amount) external
returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "BEP20:
transfer amount exceeds allowance"));
    return true;
  }
```

CertiK Label

```
   /*@CTK "BEP20Token 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 29 | BEP20Token increaseAllowance

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	increaseAllowance

Contract Code

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

CertiK Label

```
   /*@CTK "BEP20Token increaseAllowance"
      @tag assume_completion
      @post __post._allowances[msg.sender][spender] == _allowances[msg.sender][spender] +
addedValue
    */
```

The code meets the specification.

Request 30 | BEP20Token decreaseAllowance

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	decreaseAllowance

Contract Code

```
  function decreaseAllowance(address spender, uint256 subtractedValue) public returns
(bool) {
    _approve(_msgSender(), spender, _allowances[_msgSender()]
[spender].sub(subtractedValue, "BEP20: decreased allowance below zero"));
    return true;
  }
```

CertiK Label

```
   /*@CTK "BEP20Token decreaseAllowance"
      @tag assume_completion
      @post __post._allowances[msg.sender][spender] == _allowances[msg.sender][spender] -
subtractedValue
    */
```

The code meets the specification.

Request 31 | BEP20Token mint

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	mint

Contract Code

```
function mint(uint256 amount) public onlyOwner returns (bool) {
  _mint(_msgSender(), amount);
  return true;
}
```

CertiK Label

```
/*@CTK "BEP20Token mint"
   @tag assume_completion
   @pre msg.sender != address(0)
   @post __post._totalSupply == _totalSupply + amount
   @post __post._balances[msg.sender] == _balances[msg.sender] + amount
 */
```

The code meets the specification.

Request 32 | BEP20Token _transfer

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	_transfer

Contract Code

```
  function _transfer(address sender, address recipient, uint256 amount) internal {
    require(sender != address(0), "BEP20: transfer from the zero address");
    require(recipient != address(0), "BEP20: transfer to the zero address");

    _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount exceeds
balance");
    _balances[recipient] = _balances[recipient].add(amount);
    emit Transfer(sender, recipient, amount);
  }
```

CertiK Label

```
   /*@CTK "BEP20Token _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 33 | BEP20Token _mint

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	_mint

Contract Code

```
function _mint(address account, uint256 amount) internal {
  require(account != address(0), "BEP20: mint to the zero address");

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

CertiK Label

```
/*@CTK "BEP20Token _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 34 | BEP20Token _burn

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	_burn

Contract Code

```
  function _burn(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: burn from the zero address");

    _balances[account] = _balances[account].sub(amount, "BEP20: burn amount exceeds
balance");
    _totalSupply = _totalSupply.sub(amount);
    emit Transfer(account, address(0), amount);
  }
```

CertiK Label

```
   /*@CTK "BEP20Token _burn"
      @tag assume_completion
      @pre account != address(0)
      @post __post._balances[account] == _balances[account] - amount
      @post __post._totalSupply == _totalSupply - amount
    */
```

The code meets the specification.

Request 35 | BEP20Token _approve

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	_approve

Contract Code

```
function _approve(address owner, address spender, uint256 amount) internal {
  require(owner != address(0), "BEP20: approve from the zero address");
  require(spender != address(0), "BEP20: approve to the zero address");

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

CertiK Label

```
/*@CTK "BEP20Token _approve"
    @pre owner != address(0)
    @pre spender != address(0)
    @post __post._allowances[owner][spender] == amount
  */
```

The code meets the specification.

Request 36 | BEP20Token _burnFrom

Status	Details	File	Contract	Method
Success	The code meets the specification.	BEP20Token.sol	BEP20Token	_burnFrom

Contract Code

```
  function _burnFrom(address account, uint256 amount) internal {
    _burn(account, amount);
    _approve(account, _msgSender(), _allowances[account][_msgSender()].sub(amount,
"BEP20: burn amount exceeds allowance"));
  }
```

CertiK Label

```
   /*@CTK "BEP20Token _burnFrom"
      @tag assume_completion
      @post account != 0x0
      @post amount <= _balances[account] && amount <= _allowances[account][msg.sender]
      @post __post._balances[account] == _balances[account] - amount
      @post __post._totalSupply == _totalSupply - amount
      @post __post._allowances[account][msg.sender] == _allowances[account][msg.sender] -
amount
     */
```

The code meets the specification.

Appendix

Finding Categories

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.