CERTIK

Audit Report

Produced by CertiK

for

March 23, 2021

CertiK Audit Report for Convergence Finance

Request Date: 2021-03-23
Revision Date: 2021-03-23
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are 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 review.

CertiK Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.

- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.

- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for Convergence Finance to discover issues and vulnerabilities in the source code of their ConvergenceToken smart contracts. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, 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.

Vulnerability Classification

CertiK categorizes issues into three buckets based on overall risk levels:

Critical

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Medium

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Low

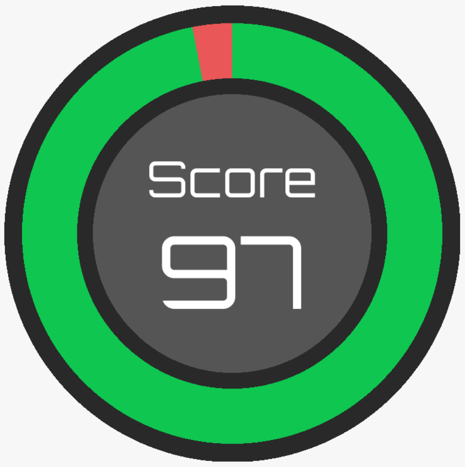
Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Mar 23, 2021



Type of Issues

CertiK's smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by miners to some degree.	0	SWC-116
Insecure Compiler Version	Using a fixed outdated compiler version or floating pragma can be problematic if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners to some degree.	0	SWC-120
"tx.origin" for Authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default, meaning a malicious user can make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized Variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Review Notes

Source Code SHA-256 Checksum

- **ConvergenceToken.sol**[1]
  `d9d086932b36df750cf8cbcbc728ccfe7d15fca441998d58472103956ef69bf7`

Summary

CertiK team is invited by The ConvergenceToken team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checkings as well as manual reviews by smart contract experts. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and ConvergenceTokenToken team has been actively giving us updates for the source code and feedback about the business logics.

Meanwhile, it is recommended to have a more well-detailed document for the public to describe the source code specifications and implementations.

Overall we found the ConvergenceToken contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrive controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is not vulnerable to any classically known antipatterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend seeking multiple opinions, more test coverage and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

ConvergenceToken.sol

- INFO Literals with many digits are difficult to read and review. Consider using scientific notation or ether suffix for variable `TOTAL_SUPPLY`.

[1] `<https://etherscan.io/address/0xc834fa996fa3bec7aad3693af486ae53d8aa8b50>`

Static Analysis Results

INSECURE_COMPILER_VERSION

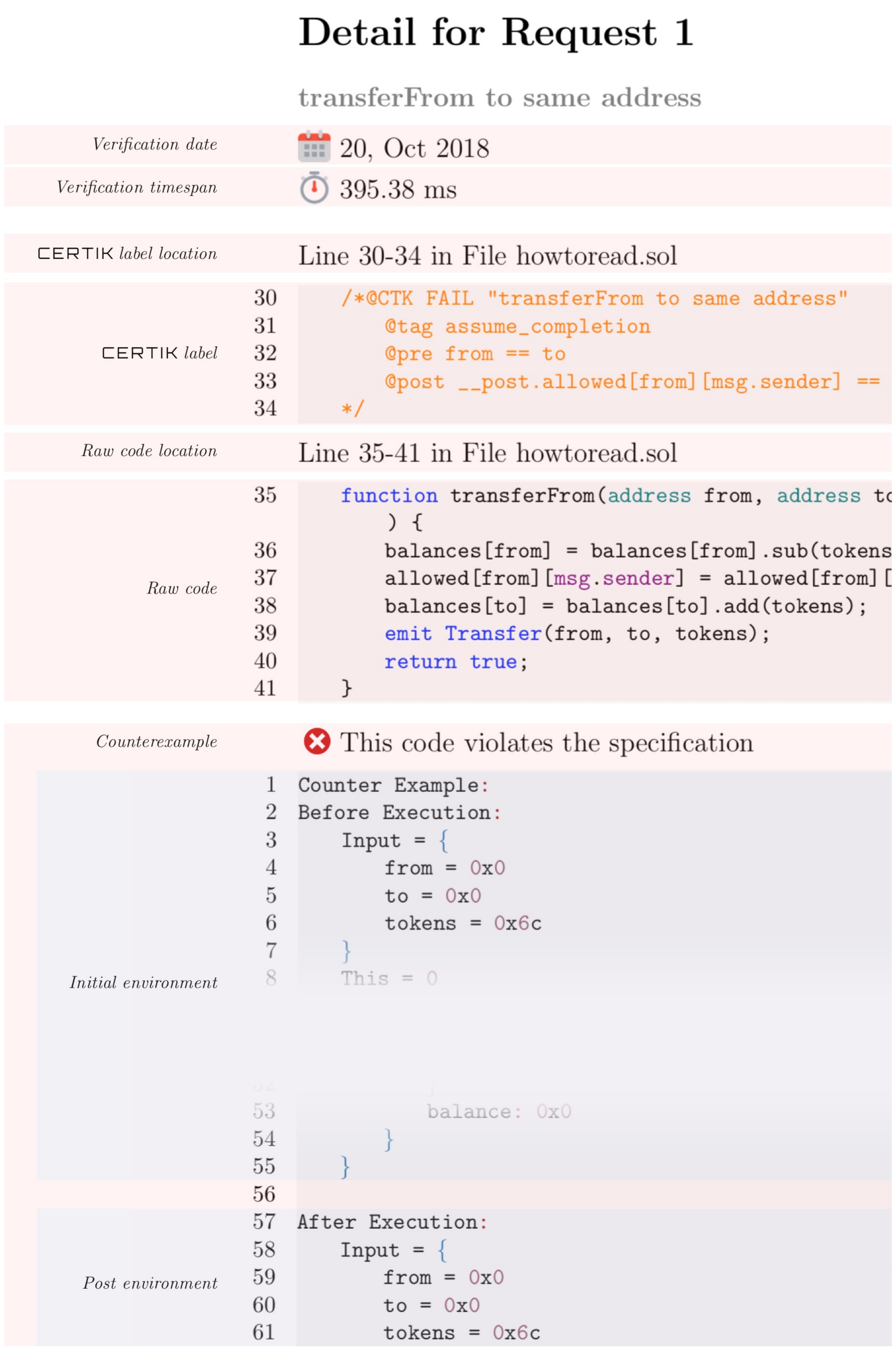
Line 9 in File ConvergenceToken.sol

```
pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

Formal Verification Results

How to read



Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms
CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location	Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address to
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample This code violates the specification

Initial environment

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            balance: 0x0
        }
    }

```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Formal Verification Request 1

Context _msgSender

📅 23, Mar 2021
⏱ 20.18 ms

Line 22-25 in File ConvergenceToken.sol

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

Line 27-29 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 2

Context _msgData

📅 23, Mar 2021
⏱ 5.48 ms

Line 31-34 in File ConvergenceToken.sol

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

Line 36-39 in File ConvergenceToken.sol

```
    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;
    }
```

✓ The code meets the specification.

Formal Verification Request 3

SafeMath tryAdd

📅 23, Mar 2021
⏱ 38.0 ms

Line 147-153 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath tryAdd"
        @tag spec
        @tag is_pure
        @post a + b < a || a + b < b -> __has_overflow
        @post a + b < a -> __return == false && __return1 == 0
        @post a + b >= a -> __return == true && __return1 == a + b
    */
```

Line 155-159 in File ConvergenceToken.sol

```
    function tryAdd(uint256 a, uint256 b) internal pure returns (bool,
↪   uint256) {
        uint256 c = a + b;
        if (c < a) return (false, 0);
        return (true, c);
    }
```

✓ The code meets the specification.

Formal Verification Request 4

SafeMath trySub

📅 23, Mar 2021
⏱ 11.55 ms

Line 167-172 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath trySub"
        @tag spec
        @tag is_pure
        @post b > a -> __return == false && __return1 == 0
        @post b <= a -> __return == true && __return1 == a - b
    */
```

Line 174-177 in File ConvergenceToken.sol

```
    function trySub(uint256 a, uint256 b) internal pure returns (bool,
↪   uint256) {
        if (b > a) return (false, 0);
        return (true, a - b);
    }
```

✓ The code meets the specification.

Formal Verification Request 5

SafeMath tryMul

📅 23, Mar 2021
⏱ 95.86 ms

Line 185-191 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath tryMul"
        @tag spec
        @tag is_pure
        @post a == 0 -> __return == true && __return1 == 0
        @post a != 0 && (a * b) / a != b -> __return == false && __return1
↪   == 0
        @post a != 0 && (a * b) / a == b -> __return == true && __return1 ==
↪   a * b
    */
```

Line 193-201 in File ConvergenceToken.sol

```
    function tryMul(uint256 a, uint256 b) internal pure returns (bool,
↪   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 (true, 0);
        uint256 c = a * b;
        if (c / a != b) return (false, 0);
        return (true, c);
    }
```

✓ The code meets the specification.

Formal Verification Request 6

SafeMath tryDiv

📅 23, Mar 2021
⏱ 12.81 ms

Line 209-214 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath tryDiv"
        @tag spec
        @tag is_pure
        @post b == 0 -> __return == false && __return1 == 0
        @post b != 0 -> __return == true && __return1 == a / b
    */
```

Line 216-219 in File ConvergenceToken.sol

```
    function tryDiv(uint256 a, uint256 b) internal pure returns (bool,
↪   uint256) {
        if (b == 0) return (false, 0);
        return (true, a / b);
    }
```

✓ The code meets the specification.

Formal Verification Request 7

SafeMath tryMod

23, Mar 2021
14.44 ms

Line 227-232 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath tryMod"
        @tag spec
        @tag is_pure
        @post b == 0 -> __return == false && __return1 == 0
        @post b != 0 -> __return == true && __return1 == a % b
    */
```

Line 234-237 in File ConvergenceToken.sol

```
    function tryMod(uint256 a, uint256 b) internal pure returns (bool,
↪   uint256) {
        if (b == 0) return (false, 0);
        return (true, a % b);
    }
```

✓ The code meets the specification.

Formal Verification Request 8

If method completes, integer overflow would not happen.

23, Mar 2021
18.96 ms

Line 250 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 261-265 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 9

SafeMath add

23, Mar 2021
4.15 ms

Line 251-259 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath add"
        @tag spec
        @tag is_pure
        @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)
    */
```

Line 261-265 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

23, Mar 2021
17.51 ms

Line 278 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 290-293 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 11

SafeMath sub

23, Mar 2021
2.27 ms

Line 279-288 in File ConvergenceToken.sol

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

Line 290-293 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 12

If method completes, integer overflow would not happen.

📅 23, Mar 2021
⏱ 55.72 ms

Line 306 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 318-323 in File ConvergenceToken.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) return 0;
        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");
        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 13

SafeMath mul

📅 23, Mar 2021
⏱ 132.1 ms

Line 307-316 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath mul"
        @tag spec
        @tag is_pure
        @tag assume_completion
        @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)
    */
```

Line 318-323 in File ConvergenceToken.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) return 0;
        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");
        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 14

If method completes, integer overflow would not happen.

📅 23, Mar 2021
⏱ 13.43 ms

Line 338 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 350-353 in File ConvergenceToken.sol

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: division by zero");
        return a / b;
    }
```

✓ The code meets the specification.

Formal Verification Request 15

SafeMath div

📅 23, Mar 2021
⏱ 2.04 ms

Line 339-348 in File ConvergenceToken.sol

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

Line 350-353 in File ConvergenceToken.sol

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: division by zero");
        return a / b;
    }
```

✓ The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.

23, Mar 2021
14.42 ms

Line 368 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 380-383 in File ConvergenceToken.sol

```
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: modulo by zero");
        return a % b;
    }
```

✓ The code meets the specification.

Formal Verification Request 17

SafeMath mod

23, Mar 2021
1.85 ms

Line 369-378 in File ConvergenceToken.sol

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

Line 380-383 in File ConvergenceToken.sol

```
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: modulo by zero");
        return a % b;
    }
```

✓ The code meets the specification.

Formal Verification Request 18

If method completes, integer overflow would not happen.

23, Mar 2021
13.94 ms

Line 399 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 411-418 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 19

SafeMath sub_with_err

23, Mar 2021
1.89 ms

Line 400-409 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath sub_with_err"
        @tag spec
        @tag is_pure
        @pre b <= a
        @post (a < b) == __reverted
        @post !__reverted -> __return == a - b
        @post !__reverted -> !__has_overflow
        @post !__reverted -> !__has_assertion_failure
        @post !(__has_buf_overflow)
    */
```

Line 411-418 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

23, Mar 2021
14.03 ms

Line 436 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 448-455 in File ConvergenceToken.sol

```
    function div(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        return a / b;
    }
```

✓ The code meets the specification.

Formal Verification Request 21

SafeMath div_with_err

23, Mar 2021
2.19 ms

Line 437-446 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath div_with_err"
        @tag spec
        @tag is_pure
        @tag assume_completion
        @post (b <= 0) == __reverted
        @post !__reverted -> __return == a / b
        @post !__reverted -> !__has_overflow
        @post !__reverted -> !__has_assertion_failure
        @post !(__has_buf_overflow)
    */
```

Line 448-455 in File ConvergenceToken.sol

```
    function div(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        return a / b;
    }
```

✓ The code meets the specification.

Formal Verification Request 22

If method completes, integer overflow would not happen.

23, Mar 2021
12.32 ms

Line 473 in File ConvergenceToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 485-492 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 23

SafeMath mod_with_err

23, Mar 2021
1.81 ms

Line 474-483 in File ConvergenceToken.sol

```
    /*@CTK "SafeMath mod_with_err"
        @tag spec
        @tag is_pure
        @tag assume_completion
        @post b != 0 -> !__reverted
        @post !__reverted -> __return == a % b
        @post !__reverted -> !__has_overflow
        @post !(__has_buf_overflow)
        @post !(__has_assertion_failure)
    */
```

Line 485-492 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 24

ERC20 constructor

📅 23, Mar 2021
⏱ 16.85 ms

Line 546-551 in File ConvergenceToken.sol

```
    /*@CTK "ERC20 constructor"
        @tag assume_completion
        @post __post._name == name_
        @post __post._symbol == symbol_
        @post __post._decimals == 18
    */
```

Line 552-556 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 25

ERC20 name

📅 23, Mar 2021
⏱ 5.71 ms

Line 562-565 in File ConvergenceToken.sol

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

Line 566-568 in File ConvergenceToken.sol

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

✅ The code meets the specification.

Formal Verification Request 26

ERC20 symbol

📅 23, Mar 2021
⏱ 7.15 ms

Line 574-577 in File ConvergenceToken.sol

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

Line 578-580 in File ConvergenceToken.sol

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

✅ The code meets the specification.

Formal Verification Request 27

ERC20 decimals

📅 23, Mar 2021
⏱ 4.6 ms

Line 596-599 in File ConvergenceToken.sol

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

Line 600-602 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 28

ERC20 totalSupply

📅 23, Mar 2021
⏱ 6.38 ms

Line 607-610 in File ConvergenceToken.sol

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

Line 611-613 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 29

ERC20 balanceOf

📅 23, Mar 2021
⏱ 7.22 ms

Line 619-622 in File ConvergenceToken.sol

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

Line 623-625 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 30

ERC20 transfer

23, Mar 2021
202.25 ms

Line 636-643 in File ConvergenceToken.sol

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

Line 644-647 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 31

ERC20 allowance

23, Mar 2021
5.11 ms

Line 653-656 in File ConvergenceToken.sol

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

Line 657-659 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 32

ERC20 approve

📅 23, Mar 2021
⏱ 70.08 ms

Line 669-673 in File ConvergenceToken.sol

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

Line 674-677 in File ConvergenceToken.sol

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

✅ The code meets the specification.

Formal Verification Request 33

ERC20 transferFrom

📅 23, Mar 2021
⏱ 364.76 ms

Line 693-703 in File ConvergenceToken.sol

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

Line 704-716 in File ConvergenceToken.sol

```
    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;
    }
```

✓ The code meets the specification.

Formal Verification Request 34

ERC20 increaseAllowance

📅 23, Mar 2021
⏱ 54.18 ms

Line 731-735 in File ConvergenceToken.sol

```
    /*@CTK "ERC20 increaseAllowance"
        @tag assume_completion
        @pre spender != address(0)
        @post __post._allowances[msg.sender][spender] ==
↪   _allowances[msg.sender][spender] + addedValue
    */
```

Line 736-739 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 35

ERC20 decreaseAllowance

📅 23, Mar 2021
⏱ 66.76 ms

Line 757-762 in File ConvergenceToken.sol

```
    /*@CTK "ERC20 decreaseAllowance"
        @tag assume_completion
        @pre spender != address(0)
        @pre subtractedValue < _allowances[msg.sender][spender]
        @post __post._allowances[msg.sender][spender] ==
↪   _allowances[msg.sender][spender] - subtractedValue
    */
```

Line 763-770 in File ConvergenceToken.sol

```
    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;
    }
```

✓ The code meets the specification.

Formal Verification Request 36

ERC20 _transfer

23, Mar 2021
75.47 ms

Line 788-796 in File ConvergenceToken.sol

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

Line 797-810 in File ConvergenceToken.sol

```
    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);
    }
```

✓ The code meets the specification.

Formal Verification Request 37

ERC20 _mint

23, Mar 2021
88.21 ms

Line 822-827 in File ConvergenceToken.sol

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

Line 828-836 in File ConvergenceToken.sol

```
    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);
    }
```

✓ The code meets the specification.

Formal Verification Request 38

ERC20 _burn

23, Mar 2021
202.18 ms

Line 850-856 in File ConvergenceToken.sol

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

Line 857-865 in File ConvergenceToken.sol

```
    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);
    }
```

✓ The code meets the specification.

Formal Verification Request 39

ERC20 _approve

📅 23, Mar 2021
⏱ 3.02 ms

Line 881-886 in File ConvergenceToken.sol

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

Line 887-897 in File ConvergenceToken.sol

```
    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);
    }
```

✓ The code meets the specification.

Formal Verification Request 40

ERC20 _setupDecimals

23, Mar 2021
6.67 ms

Line 907-910 in File ConvergenceToken.sol

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

Line 911-913 in File ConvergenceToken.sol

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

✓ The code meets the specification.

Formal Verification Request 41

ConvergenceToken constructor

23, Mar 2021
152.77 ms

Line 949-954 in File ConvergenceToken.sol

```
    /*@CTK "ConvergenceToken constructor"
        @tag assume_completion
        @pre genesis_holder != address(0)
        @post __post._totalSupply == _totalSupply + TOTAL_SUPPLY
        @post __post._balances[genesis_holder] == _balances[genesis_holder]
↪   + TOTAL_SUPPLY
    */
```

Line 955-958 in File ConvergenceToken.sol

```
    constructor(address genesis_holder) {
        require(genesis_holder != address(0), "ConvergenceToken: zero
↪   address");
        _mint(genesis_holder, TOTAL_SUPPLY);
    }
```

✓ The code meets the specification.

Source Code with CertiK Labels

ConvergenceToken.sol

```
/**
 *Submitted for verification at Etherscan.io on 2021-03-22
*/

// SPDX-License-Identifier: MIT

// File @openzeppelin/contracts/utils/Context.sol@v3.4.1

pragma solidity >=0.6.0 <0.8.0;

/*
 * @dev Provides information about the current execution context, including
↪  the
 * sender of the transaction and its data. While these are generally
↪  available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending
↪  and
 * paying for execution may not be the actual sender (as far as an
↪  application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    /*@CTK "Context _msgSender"
        @tag assume_completion
        @post __return == msg.sender
    */
    /* CertiK Smart Labelling, for more details visit: https://certik.org */
    function _msgSender() internal view virtual returns (address payable) {
        return msg.sender;
    }

    /*@CTK "Context _msgData"
        @tag assume_completion
        @post __return == msg.data
    */
    /* CertiK Smart Labelling, for more details visit: https://certik.org */
    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;
    }
}

// File @openzeppelin/contracts/token/ERC20/IERC20.sol@v3.4.1



/**
 * @title ConvergenceToken
 *
 * @dev A minimal ERC20 token contract for the Convergence token.
 */
contract ConvergenceToken is ERC20("Convergence", "CONV") {
    uint256 private constant TOTAL_SUPPLY = 10000000000e18;


    /*@CTK "ConvergenceToken constructor"
        @tag assume_completion
        @pre genesis_holder != address(0)
        @post __post._totalSupply == _totalSupply + TOTAL_SUPPLY
        @post __post._balances[genesis_holder] == _balances[genesis_holder]
 ↪  + TOTAL_SUPPLY
    */
    constructor(address genesis_holder) {
        require(genesis_holder != address(0), "ConvergenceToken: zero
 ↪  address");
        _mint(genesis_holder, TOTAL_SUPPLY);
    }
}
```