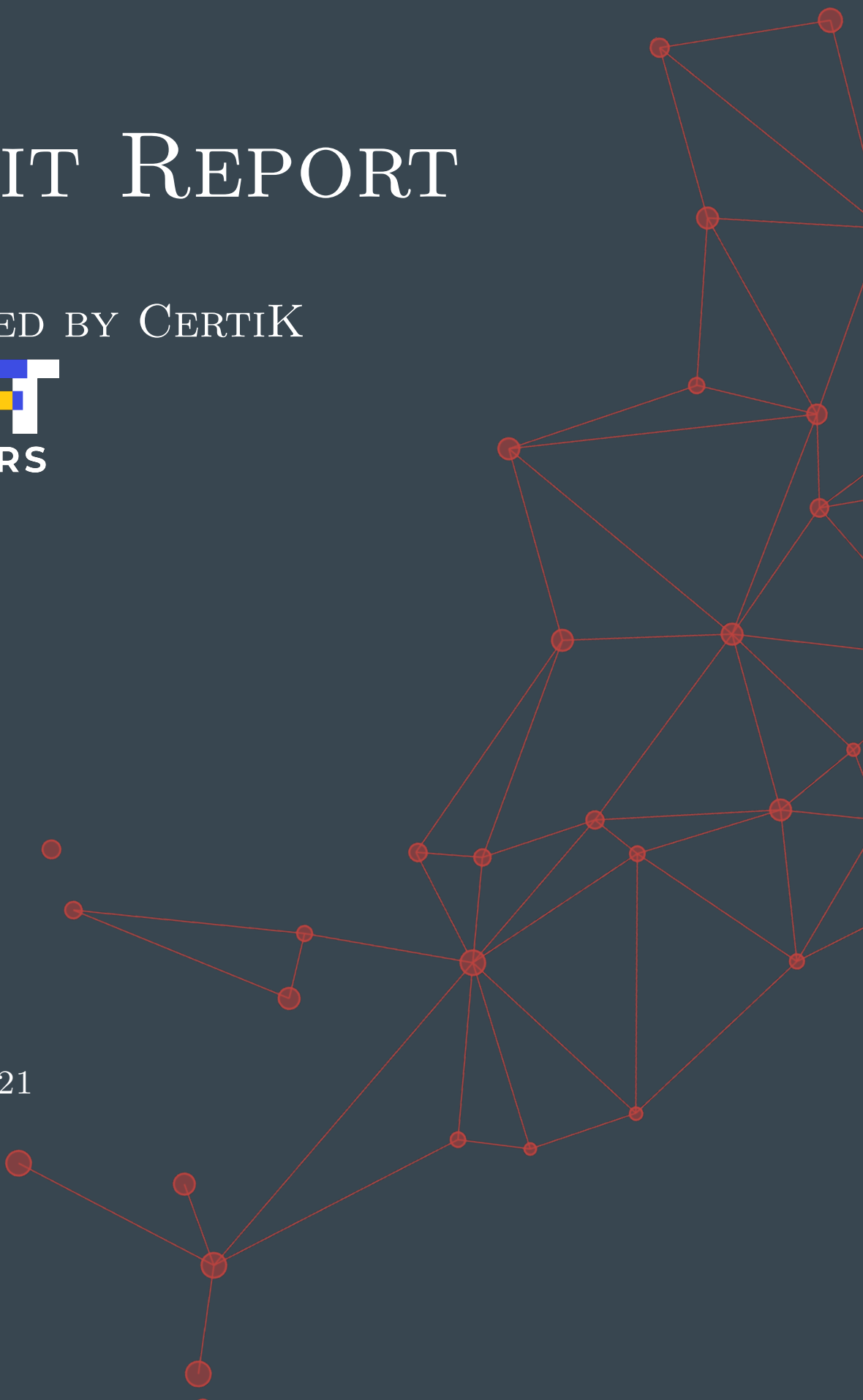
CERTIK

# Audit Report

## Produced by CertiK

## for NFT STARS

May 19, 2021



# CertiK Audit Report For NFT Stars


Request Date: 2021-05-19
Revision Date: 2021-05-19
Platform Name: Ethereum

CERTIK

# Contents

# 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 Verification Services Agreement between CertiK and NFT Stars (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, 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.

# 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, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.io/

# Executive Summary

This report has been prepared for NFT Stars to discover issues and vulnerabilities in the source code of their nftstoken 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.*

*May 19, 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

- **nftstoken.sol**
  `6574c99033e49ec4ad693d68c02b149b0c094e17e6450a0964eaa3a1430f45a9`

## Summary

CertiK worked closely with NFT Stars to audit the design and implementation of its soon-to-be released smart contract. To ensure comprehensive protection, the source code was analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. 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 best practices.

Our client NFT Stars has demonstrated their professional and knowledgeable understanding of the project NFT Stars, by having 1) a production ready repository with high-quality source code; 2) unit tests covering the majority of its business scenarios; 3) accessible, clean, and accurate readme documents for intentions, functionalities, and responsibilities of the smart contracts.

Overall, we found NFT Stars's smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, continually improve the codebase, and perform additional tests before the mainnet release.

# Static Analysis Results

## INSECURE_COMPILER_VERSION

Line 5 in File nftstoken.sol

```
5 pragma solidity 0.6.12;
```

⚠ Version to compile has the following bug:
0.6.12: EmptyByteArrayCopy, DynamicArrayCleanup

# 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*:

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53             balance: 0x0
54         }
55     }
```

*Post environment*:

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

## Formal Verification Request 1

Context___msgSender

19, May 2021
5.46 ms

Line 18-20 in File nftstoken.sol

```
18      /*@CTK Context__msgSender
19          @post __return == msg.sender
20      */
```

Line 21-23 in File nftstoken.sol

```
21      function _msgSender() internal view virtual returns (address payable) {
22          return msg.sender;
23      }
```

The code meets the specification.

## Formal Verification Request 2

Context___msgData

19, May 2021
7.05 ms

Line 25-27 in File nftstoken.sol

```
25      /*@CTK Context__msgData
26          @post __return == msg.data
27      */
```

Line 28-31 in File nftstoken.sol

```
28      function _msgData() internal view virtual returns (bytes memory) {
29          this; // silence state mutability warning without generating
    ↪ bytecode - see https://github.com/ethereum/solidity/issues/2691
30          return msg.data;
31      }
```

The code meets the specification.

## Formal Verification Request 3

SafeMath_add

19, May 2021
18.55 ms

Line 132-137 in File nftstoken.sol

```
132     /*@CTK SafeMath_add
133         @tag assume_completion
134         @tag is_pure
135         @post (a+b < a) == __reverted
136         @post !__reverted -> __return == a+b
137     */
```

Line 138-143 in File nftstoken.sol

```
138     function add(uint256 a, uint256 b) internal pure returns (uint256) {
139         uint256 c = a + b;
140         require(c >= a, "SafeMath: addition overflow");
141 
142         return c;
143     }
```

The code meets the specification.

## Formal Verification Request 4

**If method completes, integer overflow would not happen.**

19, May 2021
46.44 ms

Line 155 in File nftstoken.sol

```
155     //@CTK NO_OVERFLOW
```

Line 162-164 in File nftstoken.sol

```
162     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
163         return sub(a, b, "SafeMath: subtraction overflow");
164     }
```

The code meets the specification.

## Formal Verification Request 5

**SafeMath_sub**

19, May 2021
2.16 ms

Line 156-161 in File nftstoken.sol

```
156     /*@CTK SafeMath_sub
157         @tag assume_completion
158         @tag is_pure
159         @post b <= a
160         @post __return == a - b
161     */
```

Line 162-164 in File nftstoken.sol

```
162     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
163         return sub(a, b, "SafeMath: subtraction overflow");
164     }
```

The code meets the specification.

## Formal Verification Request 6

**If method completes, integer overflow would not happen.**

19, May 2021
1.93 ms

Line 176 in File nftstoken.sol

```
176     //@CTK NO_OVERFLOW
```

Line 183-188 in File nftstoken.sol

```
183     function sub(uint256 a, uint256 b, string memory errorMessage) internal
    ↪ pure returns (uint256) {
184         require(b <= a, errorMessage);
185         uint256 c = a - b;
186 
187         return c;
188     }
```

The code meets the specification.

## Formal Verification Request 7

**SafeMath_sub**

19, May 2021
2.89 ms

Line 177-182 in File nftstoken.sol

```
177     /*@CTK SafeMath_sub
178         @tag assume_completion
179         @tag is_pure
180         @post b <= a
181         @post __return == a - b
182     */
```

Line 183-188 in File nftstoken.sol

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

```
185         uint256 c = a - b;
186
187         return c;
188     }
```

The code meets the specification.

## Formal Verification Request 8

**If method completes, integer overflow would not happen.**

19, May 2021
34.94 ms

Line 200 in File nftstoken.sol

```
200     //@CTK NO_OVERFLOW
```

Line 208-220 in File nftstoken.sol

```
208     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
209         // Gas optimization: this is cheaper than requiring 'a' not being
    ↪  zero, but the
210         // benefit is lost if 'b' is also tested.
211         // See:
    ↪  https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
212         if (a == 0) {
213             return 0;
214         }
215
216         uint256 c = a * b;
217         require(c / a == b, "SafeMath: multiplication overflow");
218
219         return c;
220     }
```

The code meets the specification.

## Formal Verification Request 9

**SafeMath_mul**

19, May 2021
126.62 ms

Line 201-207 in File nftstoken.sol

```
201     /*@CTK SafeMath_mul
202         @tag assume_completion
203         @tag is_pure
204         @post a==0 -> __return == 0
```

```
205         @post a!=0 -> ((a>0) && (a*b/a != b)) == (__reverted)
206         @post !__reverted -> __return == a * b
207     */
```

Line 208-220 in File nftstoken.sol

```
208     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
209         // Gas optimization: this is cheaper than requiring 'a' not being
    ↪  zero, but the
210         // benefit is lost if 'b' is also tested.
211         // See:
    ↪  https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
212         if (a == 0) {
213             return 0;
214         }
215 
216         uint256 c = a * b;
217         require(c / a == b, "SafeMath: multiplication overflow");
218 
219         return c;
220     }
```

The code meets the specification.

## Formal Verification Request 10

**If method completes, integer overflow would not happen.**

19, May 2021
58.37 ms

Line 234 in File nftstoken.sol

```
234     //@CTK NO_OVERFLOW
```

Line 241-243 in File nftstoken.sol

```
241     function div(uint256 a, uint256 b) internal pure returns (uint256) {
242         return div(a, b, "SafeMath: division by zero");
243     }
```

The code meets the specification.

## Formal Verification Request 11

**SafeMath_div**

19, May 2021
4.02 ms

Line 235-240 in File nftstoken.sol

```
235     /*@CTK SafeMath_div
236         @tag assume_completion
237         @tag is_pure
238         @post b > 0
239         @post __return == a / b
240     */
```

Line 241-243 in File nftstoken.sol

```
241     function div(uint256 a, uint256 b) internal pure returns (uint256) {
242         return div(a, b, "SafeMath: division by zero");
243     }
```

The code meets the specification.

## Formal Verification Request 12

**If method completes, integer overflow would not happen.**

19, May 2021
2.84 ms

Line 257 in File nftstoken.sol

```
257     //@CTK NO_OVERFLOW
```

Line 264-270 in File nftstoken.sol

```
264     function div(uint256 a, uint256 b, string memory errorMessage) internal
    ↪   pure returns (uint256) {
265         require(b > 0, errorMessage);
266         uint256 c = a / b;
267         // assert(a == b * c + a % b); // There is no case in which this
    ↪   doesn't hold
268
269         return c;
270     }
```

The code meets the specification.

## Formal Verification Request 13

**SafeMath_div**

19, May 2021
2.36 ms

Line 258-263 in File nftstoken.sol

```
258     /*@CTK SafeMath_div
259         @tag assume_completion
260         @tag is_pure
```

```
261         @post b > 0
262         @post __return == a / b
263     */
```

Line 264-270 in File nftstoken.sol

```
264     function div(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
265         require(b > 0, errorMessage);
266         uint256 c = a / b;
267         // assert(a == b * c + a % b); // There is no case in which this
    ↪  doesn't hold
268 
269         return c;
270     }
```

✅ The code meets the specification.

## Formal Verification Request 14

**If method completes, integer overflow would not happen.**

📅 19, May 2021
⏱ 47.2 ms

Line 284 in File nftstoken.sol

```
284     //@CTK NO_OVERFLOW
```

Line 291-293 in File nftstoken.sol

```
291     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
292         return mod(a, b, "SafeMath: modulo by zero");
293     }
```

✅ The code meets the specification.

## Formal Verification Request 15

**SafeMath_mod**

📅 19, May 2021
⏱ 1.79 ms

Line 285-290 in File nftstoken.sol

```
285     /*@CTK SafeMath_mod
286         @tag assume_completion
287         @tag is_pure
288         @post b != 0
289         @post __return == a % b
290     */
```

Line 291-293 in File nftstoken.sol

```
291     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
292         return mod(a, b, "SafeMath: modulo by zero");
293     }
```

The code meets the specification.

## Formal Verification Request 16

**If method completes, integer overflow would not happen.**

19, May 2021
0.95 ms

Line 307 in File nftstoken.sol

```
307     //@CTK NO_OVERFLOW
```

Line 314-317 in File nftstoken.sol

```
314     function mod(uint256 a, uint256 b, string memory errorMessage) internal
    ↪ pure returns (uint256) {
315         require(b != 0, errorMessage);
316         return a % b;
317     }
```

The code meets the specification.

## Formal Verification Request 17

**SafeMath_mod**

19, May 2021
1.94 ms

Line 308-313 in File nftstoken.sol

```
308     /*@CTK SafeMath_mod
309         @tag assume_completion
310         @tag is_pure
311         @post b != 0
312         @post __return == a % b
313     */
```

Line 314-317 in File nftstoken.sol

```
314     function mod(uint256 a, uint256 b, string memory errorMessage) internal
    ↪ pure returns (uint256) {
315         require(b != 0, errorMessage);
316         return a % b;
317     }
```

The code meets the specification.

## Formal Verification Request 18

ERC20_constructor

19, May 2021
14.66 ms

Line 509-513 in File nftstoken.sol

```
509     /*@CTK ERC20_constructor
510         @post __post._name == name
511         @post __post._symbol == symbol
512         @post __post._decimals == 18
513     */
```

Line 514-518 in File nftstoken.sol

```
514     constructor (string memory name, string memory symbol) public {
515         _name = name;
516         _symbol = symbol;
517         _decimals = 18;
518     }
```

The code meets the specification.

## Formal Verification Request 19

ERC20_name

19, May 2021
5.09 ms

Line 523-525 in File nftstoken.sol

```
523     /*@CTK ERC20_name
524         @post __return == _name
525     */
```

Line 526-528 in File nftstoken.sol

```
526     function name() public view returns (string memory) {
527         return _name;
528     }
```

The code meets the specification.

## Formal Verification Request 20

ERC20_symbol

19, May 2021
7.33 ms

Line 534-536 in File nftstoken.sol

```
534     /*@CTK ERC20_symbol
535         @post __return == _symbol
536     */
```

Line 537-539 in File nftstoken.sol

```
537     function symbol() public view returns (string memory) {
538         return _symbol;
539     }
```

✅ The code meets the specification.

## Formal Verification Request 21

ERC20_decimals

📅 19, May 2021
⏱ 4.26 ms

Line 554-556 in File nftstoken.sol

```
554     /*@CTK ERC20_decimals
555         @post __return == _decimals
556     */
```

Line 557-559 in File nftstoken.sol

```
557     function decimals() public view returns (uint8) {
558         return _decimals;
559     }
```

✅ The code meets the specification.

## Formal Verification Request 22

ERC20_totalSupply

📅 19, May 2021
⏱ 5.41 ms

Line 564-566 in File nftstoken.sol

```
564     /*@CTK ERC20_totalSupply
565         @post __return == _totalSupply
566     */
```

Line 567-569 in File nftstoken.sol

```
567     function totalSupply() public view override returns (uint256) {
568         return _totalSupply;
569     }
```

✅ The code meets the specification.

## Formal Verification Request 23

ERC20_balanceOf

19, May 2021
7.0 ms

Line 574-576 in File nftstoken.sol

```
574     /*@CTK ERC20_balanceOf
575         @post __return == _balances[account]
576     */
```

Line 577-579 in File nftstoken.sol

```
577     function balanceOf(address account) public view override returns
    ↪   (uint256) {
578         return _balances[account];
579     }
```

The code meets the specification.

## Formal Verification Request 24

ERC20_transfer

19, May 2021
325.2 ms

Line 589-596 in File nftstoken.sol

```
589     /*@CTK ERC20_transfer
590         @tag assume_completion
591         @post recipient != address(0)
592         @post (_balances[msg.sender] < amount) == (__reverted)
593         @post (!__reverted && msg.sender != recipient) ->
    ↪   (__post._balances[msg.sender] == _balances[msg.sender] - amount)
594         @post (!__reverted && msg.sender != recipient) ->
    ↪   (__post._balances[recipient] == _balances[recipient] + amount)
595         @post (!__reverted && msg.sender == recipient) ->
    ↪   (__post._balances[msg.sender] == _balances[msg.sender])
596     */
```

Line 597-600 in File nftstoken.sol

```
597     function transfer(address recipient, uint256 amount) public virtual
    ↪   override returns (bool) {
598         _transfer(_msgSender(), recipient, amount);
599         return true;
600     }
```

The code meets the specification.

## Formal Verification Request 25

ERC20_allowance

19, May 2021
5.02 ms

Line 605-607 in File nftstoken.sol

```
605     /*@CTK ERC20_allowance
606         @post __return == _allowances[owner][spender]
607     */
```

Line 608-610 in File nftstoken.sol

```
608     function allowance(address owner, address spender) public view virtual
    ↪ override returns (uint256) {
609         return _allowances[owner][spender];
610     }
```

The code meets the specification.

## Formal Verification Request 26

ERC20_approve

19, May 2021
72.24 ms

Line 619-623 in File nftstoken.sol

```
619     /*@CTK ERC20_approve
620         @tag assume_completion
621         @post spender != address(0)
622         @post __post._allowances[msg.sender][spender] == amount
623     */
```

Line 624-627 in File nftstoken.sol

```
624     function approve(address spender, uint256 amount) public virtual override
    ↪ returns (bool) {
625         _approve(_msgSender(), spender, amount);
626         return true;
627     }
```

The code meets the specification.

## Formal Verification Request 27

ERC20_transferFrom

19, May 2021
450.6 ms

Line 641-650 in File nftstoken.sol

```
641     /*@CTK ERC20_transferFrom
642         @tag assume_completion
643         @post sender != address(0)
644         @post recipient != address(0)
645         @post (_balances[sender] < amount || _allowances[sender][msg.sender]
    ↪  < amount) == (__reverted)
646         @post (!__reverted && sender != recipient) ->
    ↪  (__post._balances[sender] == _balances[sender] - amount)
647         @post (!__reverted && sender != recipient) ->
    ↪  (__post._balances[recipient] == _balances[recipient] + amount)
648         @post (!__reverted && sender == recipient) ->
    ↪  (__post._balances[sender] == _balances[sender])
649         @post (!__reverted) -> (__post._allowances[sender][msg.sender] ==
    ↪  _allowances[sender][msg.sender] - amount)
650     */
```

Line 651-655 in File nftstoken.sol

```
651     function transferFrom(address sender, address recipient, uint256 amount)
    ↪  public virtual override returns (bool) {
652         _transfer(sender, recipient, amount);
653         _approve(sender, _msgSender(),
    ↪  _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount
    ↪  exceeds allowance"));
654         return true;
655     }
```

The code meets the specification.

## Formal Verification Request 28

**ERC20_increaseAllowance**

19, May 2021
65.51 ms

Line 669-673 in File nftstoken.sol

```
669     /*@CTK ERC20_increaseAllowance
670         @tag assume_completion
671         @post spender != address(0)
672         @post __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] + addedValue
673     */
```

Line 674-677 in File nftstoken.sol

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

```
676         return true;
677     }
```

The code meets the specification.

## Formal Verification Request 29

ERC20_decreaseAllowance

19, May 2021
72.19 ms

Line 693-698 in File nftstoken.sol

```
693     /*@CTK ERC20_decreaseAllowance
694         @tag assume_completion
695         @post spender != address(0)
696         @post (_allowances[msg.sender][spender] < subtractedValue) ==
    ↪  __reverted
697         @post !__reverted -> __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] - subtractedValue
698     */
```

Line 699-702 in File nftstoken.sol

```
699     function decreaseAllowance(address spender, uint256 subtractedValue)
    ↪  public virtual returns (bool) {
700         _approve(_msgSender(), spender,
    ↪  _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased
    ↪  allowance below zero"));
701         return true;
702     }
```

The code meets the specification.

## Formal Verification Request 30

ERC20___transfer

19, May 2021
79.15 ms

Line 718-726 in File nftstoken.sol

```
718     /*@CTK ERC20__transfer
719         @tag assume_completion
720         @post sender != address(0)
721         @post recipient != address(0)
722         @post (_balances[sender] < amount) == (__reverted)
723         @post (!__reverted && sender != recipient) ->
    ↪  (__post._balances[sender] == _balances[sender] - amount)
```

```
724         @post (!__reverted && sender != recipient) ->
    ↪ (__post._balances[recipient] == _balances[recipient] + amount)
725         @post (!__reverted && sender == recipient) ->
    ↪ (__post._balances[sender] == _balances[sender])
726     */
```

Line 727-736 in File nftstoken.sol

```
727     function _transfer(address sender, address recipient, uint256 amount)
    ↪ internal virtual {
728         require(sender != address(0), "ERC20: transfer from the zero
    ↪ address");
729         require(recipient != address(0), "ERC20: transfer to the zero
    ↪ address");
730 
731         _beforeTokenTransfer(sender, recipient, amount);
732 
733         _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer
    ↪ amount exceeds balance");
734         _balances[recipient] = _balances[recipient].add(amount);
735         emit Transfer(sender, recipient, amount);
736     }
```

The code meets the specification.

## Formal Verification Request 31

**ERC20___mint**

19, May 2021
97.89 ms

Line 747-752 in File nftstoken.sol

```
747     /*@CTK ERC20__mint
748         @tag assume_completion
749         @post account != address(0)
750         @post __post._totalSupply == _totalSupply + amount
751         @post __post._balances[account] == _balances[account] + amount
752     */
```

Line 753-761 in File nftstoken.sol

```
753     function _mint(address account, uint256 amount) internal virtual {
754         require(account != address(0), "ERC20: mint to the zero address");
755 
756         _beforeTokenTransfer(address(0), account, amount);
757 
758         _totalSupply = _totalSupply.add(amount);
759         _balances[account] = _balances[account].add(amount);
760         emit Transfer(address(0), account, amount);
761     }
```

The code meets the specification.

## Formal Verification Request 32

**ERC20___burn**

19, May 2021
167.15 ms

Line 774-780 in File nftstoken.sol

```
774     /*@CTK ERC20__burn
775         @tag assume_completion
776         @post account != address(0)
777         @post (_balances[account] < amount) == (__reverted)
778         @post (!__reverted) -> (__post._balances[account] ==
    ↪  _balances[account] - amount)
779         @post (!__reverted) -> (__post._totalSupply == _totalSupply -
    ↪  amount)
780     */
```

Line 781-789 in File nftstoken.sol

```
781     function _burn(address account, uint256 amount) internal virtual {
782         require(account != address(0), "ERC20: burn from the zero address");
783 
784         _beforeTokenTransfer(account, address(0), amount);
785 
786         _balances[account] = _balances[account].sub(amount, "ERC20: burn
    ↪  amount exceeds balance");
787         _totalSupply = _totalSupply.sub(amount);
788         emit Transfer(account, address(0), amount);
789     }
```

The code meets the specification.

## Formal Verification Request 33

**ERC20___approve**

19, May 2021
3.01 ms

Line 804-809 in File nftstoken.sol

```
804     /*@CTK ERC20__approve
805         @tag assume_completion
806         @post owner != address(0)
807         @post spender != address(0)
808         @post __post._allowances[owner][spender] == amount
809     */
```

Line 810-816 in File nftstoken.sol

```
810     function _approve(address owner, address spender, uint256 amount)
    ↪   internal virtual {
811         require(owner != address(0), "ERC20: approve from the zero address");
812         require(spender != address(0), "ERC20: approve to the zero address");
813 
814         _allowances[owner][spender] = amount;
815         emit Approval(owner, spender, amount);
816     }
```

The code meets the specification.

## Formal Verification Request 34

ERC20___setupDecimals

19, May 2021
6.39 ms

Line 825-827 in File nftstoken.sol

```
825     /*@CTK ERC20__setupDecimals
826         @post __post._decimals == decimals_
827     */
```

Line 828-830 in File nftstoken.sol

```
828     function _setupDecimals(uint8 decimals_) internal {
829         _decimals = decimals_;
830     }
```

The code meets the specification.

## Formal Verification Request 35

Ownable__constructor

19, May 2021
20.99 ms

Line 869-871 in File nftstoken.sol

```
869     /*@CTK "Ownable_constructor"
870       @post __post._owner == msg.sender
871      */
```

Line 872-876 in File nftstoken.sol

```
872     constructor () internal {
873         address msgSender = _msgSender();
874         _owner = msgSender;
875         emit OwnershipTransferred(address(0), msgSender);
876     }
```

The code meets the specification.

## Formal Verification Request 36

Ownable_owner

19, May 2021
4.45 ms

Line 881-883 in File nftstoken.sol

```
881     /*@CTK Ownable_owner
882         @post __return == _owner
883     */
```

Line 884-886 in File nftstoken.sol

```
884     function owner() public view returns (address) {
885         return _owner;
886     }
```

The code meets the specification.

## Formal Verification Request 37

Ownable_renounceOwnership

19, May 2021
26.21 ms

Line 903-907 in File nftstoken.sol

```
903     /*@CTK "Ownable_renounceOwnership"
904       @tag assume_completion
905       @post _owner == msg.sender
906       @post __post._owner == address(0)
907      */
```

Line 908-911 in File nftstoken.sol

```
908     function renounceOwnership() public virtual onlyOwner {
909         emit OwnershipTransferred(_owner, address(0));
910         _owner = address(0);
911     }
```

The code meets the specification.

## Formal Verification Request 38

Ownable_transferOwnership

19, May 2021
36.57 ms

Line 917-922 in File nftstoken.sol

```
917     /*@CTK "Ownable_transferOwnership"
918       @tag assume_completion
919       @post _owner == msg.sender
920       @post newOwner != address(0)
921       @post __post._owner == newOwner
922      */
```

Line 923-927 in File nftstoken.sol

```
923     function transferOwnership(address newOwner) public virtual onlyOwner {
924         require(newOwner != address(0), "Ownable: new owner is the zero
    ↪  address");
925         emit OwnershipTransferred(_owner, newOwner);
926         _owner = newOwner;
927     }
```

The code meets the specification.

# Source Code with CertiK Labels

## nftstoken.sol

```
/**
 *Submitted for verification at Etherscan.io on 2021-04-28
*/

pragma solidity 0.6.12;

/*
 * @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
        @post __return == msg.sender
    */
    function _msgSender() internal view virtual returns (address payable) {
        return msg.sender;
    }

    /*@CTK Context__msgData
        @post __return == msg.data
    */
    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;
    }
}

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
```

```
40      */
41     function totalSupply() external view returns (uint256);
42
43     /**
44      * @dev Returns the amount of tokens owned by `account`.
45      */
46     function balanceOf(address account) external view returns (uint256);
47
48     /**
49      * @dev Moves `amount` tokens from the caller's account to `recipient`.
50      *
51      * Returns a boolean value indicating whether the operation succeeded.
52      *
53      * Emits a {Transfer} event.
54      */
55     function transfer(address recipient, uint256 amount) external returns
    ↪  (bool);
56
57     /**
58      * @dev Returns the remaining number of tokens that `spender` will be
59      * allowed to spend on behalf of `owner` through {transferFrom}. This is
60      * zero by default.
61      *
62      * This value changes when {approve} or {transferFrom} are called.
63      */
64     function allowance(address owner, address spender) external view returns
    ↪  (uint256);
65
66     /**
67      * @dev Sets `amount` as the allowance of `spender` over the caller's
    ↪  tokens.
68      *
69      * Returns a boolean value indicating whether the operation succeeded.
70      *
71      * IMPORTANT: Beware that changing an allowance with this method brings
    ↪  the risk
72      * that someone may use both the old and the new allowance by
    ↪  unfortunate
73      * transaction ordering. One possible solution to mitigate this race
74      * condition is to first reduce the spender's allowance to 0 and set the
75      * desired value afterwards:
76      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
77      *
78      * Emits an {Approval} event.
79      */
80     function approve(address spender, uint256 amount) external returns
    ↪  (bool);
81
```

```
82      /**
83       * @dev Moves `amount` tokens from `sender` to `recipient` using the
84       * allowance mechanism. `amount` is then deducted from the caller's
85       * allowance.
86       *
87       * Returns a boolean value indicating whether the operation succeeded.
88       *
89       * Emits a {Transfer} event.
90       */
91      function transferFrom(address sender, address recipient, uint256 amount)
    ↪   external returns (bool);
92
93      /**
94       * @dev Emitted when `value` tokens are moved from one account (`from`)
    ↪   to
95       * another (`to`).
96       *
97       * Note that `value` may be zero.
98       */
99      event Transfer(address indexed from, address indexed to, uint256 value);
100
101     /**
102      * @dev Emitted when the allowance of a `spender` for an `owner` is set
    ↪   by
103      * a call to {approve}. `value` is the new allowance.
104      */
105     event Approval(address indexed owner, address indexed spender, uint256
    ↪   value);
106 }
107
108 /**
109  * @dev Wrappers over Solidity's arithmetic operations with added overflow
110  * checks.
111  *
112  * Arithmetic operations in Solidity wrap on overflow. This can easily
    ↪   result
113  * in bugs, because programmers usually assume that an overflow raises an
114  * error, which is the standard behavior in high level programming
    ↪   languages.
115  * `SafeMath` restores this intuition by reverting the transaction when an
116  * operation overflows.
117  *
118  * Using this library instead of the unchecked operations eliminates an
    ↪   entire
119  * class of bugs, so it's recommended to use it always.
120  */
121 library SafeMath {
122     /**
```

```
123         * @dev Returns the addition of two unsigned integers, reverting on
124         * overflow.
125         *
126         * Counterpart to Solidity's `+` operator.
127         *
128         * Requirements:
129         *
130         * - Addition cannot overflow.
131         */
132        /*@CTK SafeMath_add
133            @tag assume_completion
134            @tag is_pure
135            @post (a+b < a) == __reverted
136            @post !__reverted -> __return == a+b
137        */
138        function add(uint256 a, uint256 b) internal pure returns (uint256) {
139            uint256 c = a + b;
140            require(c >= a, "SafeMath: addition overflow");
141
142            return c;
143        }
144
145        /**
146         * @dev Returns the subtraction of two unsigned integers, reverting on
147         * overflow (when the result is negative).
148         *
149         * Counterpart to Solidity's `-` operator.
150         *
151         * Requirements:
152         *
153         * - Subtraction cannot overflow.
154         */
155        //@CTK NO_OVERFLOW
156        /*@CTK SafeMath_sub
157            @tag assume_completion
158            @tag is_pure
159            @post b <= a
160            @post __return == a - b
161        */
162        function sub(uint256 a, uint256 b) internal pure returns (uint256) {
163            return sub(a, b, "SafeMath: subtraction overflow");
164        }
165
166        /**
167         * @dev Returns the subtraction of two unsigned integers, reverting with
    ↪  custom message on
168         * overflow (when the result is negative).
169         *
```

```
170      * Counterpart to Solidity's `-` operator.
171      *
172      * Requirements:
173      *
174      * - Subtraction cannot overflow.
175      */
176     //@CTK NO_OVERFLOW
177     /*@CTK SafeMath_sub
178         @tag assume_completion
179         @tag is_pure
180         @post b <= a
181         @post __return == a - b
182     */
183     function sub(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
184         require(b <= a, errorMessage);
185         uint256 c = a - b;
186 
187         return c;
188     }
189 
190     /**
191      * @dev Returns the multiplication of two unsigned integers, reverting
    ↪  on
192      * overflow.
193      *
194      * Counterpart to Solidity's `*` operator.
195      *
196      * Requirements:
197      *
198      * - Multiplication cannot overflow.
199      */
200     //@CTK NO_OVERFLOW
201     /*@CTK SafeMath_mul
202         @tag assume_completion
203         @tag is_pure
204         @post a==0 -> __return == 0
205         @post a!=0 -> ((a>0) && (a*b/a != b)) == (__reverted)
206         @post !__reverted -> __return == a * b
207     */
208     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
209         // Gas optimization: this is cheaper than requiring 'a' not being
    ↪  zero, but the
210         // benefit is lost if 'b' is also tested.
211         // See:
    ↪  https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
212         if (a == 0) {
213             return 0;
```

```
214         }
215
216         uint256 c = a * b;
217         require(c / a == b, "SafeMath: multiplication overflow");
218
219         return c;
220     }
221
222     /**
223      * @dev Returns the integer division of two unsigned integers. Reverts
    ↪  on
224      * division by zero. The result is rounded towards zero.
225      *
226      * Counterpart to Solidity's `/` operator. Note: this function uses a
227      * `revert` opcode (which leaves remaining gas untouched) while Solidity
228      * uses an invalid opcode to revert (consuming all remaining gas).
229      *
230      * Requirements:
231      *
232      * - The divisor cannot be zero.
233      */
234     //@CTK NO_OVERFLOW
235     /*@CTK SafeMath_div
236         @tag assume_completion
237         @tag is_pure
238         @post b > 0
239         @post __return == a / b
240     */
241     function div(uint256 a, uint256 b) internal pure returns (uint256) {
242         return div(a, b, "SafeMath: division by zero");
243     }
244
245     /**
246      * @dev Returns the integer division of two unsigned integers. Reverts
    ↪  with custom message on
247      * division by zero. The result is rounded towards zero.
248      *
249      * Counterpart to Solidity's `/` operator. Note: this function uses a
250      * `revert` opcode (which leaves remaining gas untouched) while Solidity
251      * uses an invalid opcode to revert (consuming all remaining gas).
252      *
253      * Requirements:
254      *
255      * - The divisor cannot be zero.
256      */
257     //@CTK NO_OVERFLOW
258     /*@CTK SafeMath_div
259         @tag assume_completion
```

```
260         @tag is_pure
261         @post b > 0
262         @post __return == a / b
263     */
264     function div(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
265         require(b > 0, errorMessage);
266         uint256 c = a / b;
267         // assert(a == b * c + a % b); // There is no case in which this
    ↪  doesn't hold
268 
269         return c;
270     }
271 
272     /**
273      * @dev Returns the remainder of dividing two unsigned integers.
    ↪  (unsigned integer modulo),
274      * Reverts when dividing by zero.
275      *
276      * Counterpart to Solidity's `%` operator. This function uses a `revert`
277      * opcode (which leaves remaining gas untouched) while Solidity uses an
278      * invalid opcode to revert (consuming all remaining gas).
279      *
280      * Requirements:
281      *
282      * - The divisor cannot be zero.
283      */
284     //@CTK NO_OVERFLOW
285     /*@CTK SafeMath_mod
286         @tag assume_completion
287         @tag is_pure
288         @post b != 0
289         @post __return == a % b
290     */
291     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
292         return mod(a, b, "SafeMath: modulo by zero");
293     }
294 
295     /**
296      * @dev Returns the remainder of dividing two unsigned integers.
    ↪  (unsigned integer modulo),
297      * Reverts with custom message when dividing by zero.
298      *
299      * Counterpart to Solidity's `%` operator. This function uses a `revert`
300      * opcode (which leaves remaining gas untouched) while Solidity uses an
301      * invalid opcode to revert (consuming all remaining gas).
302      *
303      * Requirements:
```

```
304         *
305         * - The divisor cannot be zero.
306         */
307        //@CTK NO_OVERFLOW
308        /*@CTK SafeMath_mod
309            @tag assume_completion
310            @tag is_pure
311            @post b != 0
312            @post __return == a % b
313        */
314        function mod(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
315            require(b != 0, errorMessage);
316            return a % b;
317        }
318    }
319
320    /**
321     * @dev Collection of functions related to the address type
322     */
323    library Address {
324        /**
325         * @dev Returns true if `account` is a contract.
326         *
327         * [IMPORTANT]
328         * ====
329         * It is unsafe to assume that an address for which this function
    ↪  returns
330         * false is an externally-owned account (EOA) and not a contract.
331         *
332         * Among others, `isContract` will return false for the following
333         * types of addresses:
334         *
335         *  - an externally-owned account
336         *  - a contract in construction
337         *  - an address where a contract will be created
338         *  - an address where a contract lived, but was destroyed
339         * ====
340         */
341        function isContract(address account) internal view returns (bool) {
342            // According to EIP-1052, 0x0 is the value returned for not-yet
    ↪  created accounts
343            // and
    ↪  0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is
    ↪  returned
344            // for accounts without code, i.e. `keccak256('')`
345            bytes32 codehash;
```

```
346         bytes32 accountHash =
    ↪ 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
347         // solhint-disable-next-line no-inline-assembly
348         assembly { codehash := extcodehash(account) }
349         return (codehash != accountHash && codehash != 0x0);
350     }
351
352     /**
353      * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
354      * `recipient`, forwarding all available gas and reverting on errors.
355      *
356      * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas
    ↪ cost
357      * of certain opcodes, possibly making contracts go over the 2300 gas
    ↪ limit
358      * imposed by `transfer`, making them unable to receive funds via
359      * `transfer`. {sendValue} removes this limitation.
360      *
361      *
    ↪ https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[D
    ↪ more].
362      *
363      * IMPORTANT: because control is transferred to `recipient`, care must
    ↪ be
364      * taken to not create reentrancy vulnerabilities. Consider using
365      * {ReentrancyGuard} or the
366      *
    ↪ https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-ch
    ↪ pattern].
367      */
368     function sendValue(address payable recipient, uint256 amount) internal {
369         require(address(this).balance >= amount, "Address: insufficient
    ↪ balance");
370
371         // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
372
373         if (success) {
374             return returndata;
375         } else {
376             // Look for revert reason and bubble it up if present
377             if (returndata.length > 0) {
378                 // The easiest way to bubble the revert reason is using
    ↪ memory via assembly
379
380                 // solhint-disable-next-line no-inline-assembly
381                 assembly {
382                     let returndata_size := mload(returndata)
383                     revert(add(32, returndata), returndata_size)
```

```
384                 }
385             } else {
386                 revert(errorMessage);
387             }
388         }
389     }
390 }
391
392 /**
393  * @dev Implementation of the {IERC20} interface.
394  *
395  * This implementation is agnostic to the way tokens are created. This means
396  * that a supply mechanism has to be added in a derived contract using
     ↪ {_mint}.
397  * For a generic mechanism see {ERC20PresetMinterPauser}.
398  *
399  * TIP: For a detailed writeup see our guide
400  *
     ↪ https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms/226[Ho
401  * to implement supply mechanisms].
402  *
403  * We have followed general OpenZeppelin guidelines: functions revert
     ↪ instead
404  * of returning `false` on failure. This behavior is nonetheless
     ↪ conventional
405  * and does not conflict with the expectations of ERC20 applications.
406  *
407  * Additionally, an {Approval} event is emitted on calls to {transferFrom}.
408  * This allows applications to reconstruct the allowance for all accounts
     ↪ just
409  * by listening to said events. Other implementations of the EIP may not
     ↪ emit
410  * these events, as it isn't required by the specification.
411  *
412  * Finally, the non-standard {decreaseAllowance} and {increaseAllowance}
413  * functions have been added to mitigate the well-known issues around
     ↪ setting
414  * allowances. See {IERC20-approve}.
415  */
416 contract ERC20 is Context, IERC20 {
417     using SafeMath for uint256;
418     using Address for address;
419
420     mapping (address => uint256) private _balances;
421
422     mapping (address => mapping (address => uint256)) private _allowances;
423
424     uint256 private _totalSupply;
```

```
425
426     string private _name;
427     string private _symbol;
428     uint8 private _decimals;
429
430     /**
431      * @dev Sets the values for {name} and {symbol}, initializes {decimals}
    ↪  with
432      * a default value of 18.
433      *
434      * To select a different value for {decimals}, use {_setupDecimals}.
435      *
436      * All three of these values are immutable: they can only be set once
    ↪  during
437      * construction.
438      */
439     /*@CTK ERC20_constructor
440         @post __post._name == name
441         @post __post._symbol == symbol
442         @post __post._decimals == 18
443     */
444     constructor (string memory name, string memory symbol) public {
445         _name = name;
446         _symbol = symbol;
447         _decimals = 18;
448     }
449
450     /**
451      * @dev Returns the name of the token.
452      */
453     /*@CTK ERC20_name
454         @post __return == _name
455     */
456     function name() public view returns (string memory) {
457         return _name;
458     }
459
460     /**
461      * @dev Returns the symbol of the token, usually a shorter version of
    ↪  the
462      * name.
463      */
464     /*@CTK ERC20_symbol
465         @post __return == _symbol
466     */
467     function symbol() public view returns (string memory) {
468         return _symbol;
469     }
```

```
470
471     /**
472      * @dev Returns the number of decimals used to get its user
    ↪ representation.
473      * For example, if `decimals` equals `2`, a balance of `505` tokens
    ↪ should
474      * be displayed to a user as `5,05` (`505 / 10 ** 2`).
475      *
476      * Tokens usually opt for a value of 18, imitating the relationship
    ↪ between
477      * Ether and Wei. This is the value {ERC20} uses, unless
    ↪ {_setupDecimals} is
478      * called.
479      *
480      * NOTE: This information is only used for _display_ purposes: it in
481      * no way affects any of the arithmetic of the contract, including
482      * {IERC20-balanceOf} and {IERC20-transfer}.
483      */
484     /*@CTK ERC20_decimals
485         @post __return == _decimals
486     */
487     function decimals() public view returns (uint8) {
488         return _decimals;
489     }
490
491     /**
492      * @dev See {IERC20-totalSupply}.
493      */
494     /*@CTK ERC20_totalSupply
495         @post __return == _totalSupply
496     */
497     function totalSupply() public view override returns (uint256) {
498         return _totalSupply;
499     }
500
501     /**
502      * @dev See {IERC20-balanceOf}.
503      */
504     /*@CTK ERC20_balanceOf
505         @post __return == _balances[account]
506     */
507     function balanceOf(address account) public view override returns
    ↪ (uint256) {
508         return _balances[account];
509     }
510
511     /**
512      * @dev See {IERC20-transfer}.
```

```
513      *
514      * Requirements:
515      *
516      * - `recipient` cannot be the zero address.
517      * - the caller must have a balance of at least `amount`.
518      */
519     /*@CTK ERC20_transfer
520         @tag assume_completion
521         @post recipient != address(0)
522         @post (_balances[msg.sender] < amount) == (__reverted)
523         @post (!__reverted && msg.sender != recipient) ->
    ↪  (__post._balances[msg.sender] == _balances[msg.sender] - amount)
524         @post (!__reverted && msg.sender != recipient) ->
    ↪  (__post._balances[recipient] == _balances[recipient] + amount)
525         @post (!__reverted && msg.sender == recipient) ->
    ↪  (__post._balances[msg.sender] == _balances[msg.sender])
526     */
527     function transfer(address recipient, uint256 amount) public virtual
    ↪  override returns (bool) {
528         _transfer(_msgSender(), recipient, amount);
529         return true;
530     }
531
532     /**
533      * @dev See {IERC20-allowance}.
534      */
535     /*@CTK ERC20_allowance
536         @post __return == _allowances[owner][spender]
537     */
538     function allowance(address owner, address spender) public view virtual
    ↪  override returns (uint256) {
539         return _allowances[owner][spender];
540     }
541
542     /**
543      * @dev See {IERC20-approve}.
544      *
545      * Requirements:
546      *
547      * - `spender` cannot be the zero address.
548      */
549     /*@CTK ERC20_approve
550         @tag assume_completion
551         @post spender != address(0)
552         @post __post._allowances[msg.sender][spender] == amount
553     */
554     function approve(address spender, uint256 amount) public virtual override
    ↪  returns (bool) {
```

```
555         _approve(_msgSender(), spender, amount);
556         return true;
557     }
558 
559     /**
560      * @dev See {IERC20-transferFrom}.
561      *
562      * Emits an {Approval} event indicating the updated allowance. This is
    ↪  not
563      * required by the EIP. See the note at the beginning of {ERC20};
564      *
565      * Requirements:
566      * - `sender` and `recipient` cannot be the zero address.
567      * - `sender` must have a balance of at least `amount`.
568      * - the caller must have allowance for ``sender``'s tokens of at least
569      * `amount`.
570      */
571     /*@CTK ERC20_transferFrom
572         @tag assume_completion
573         @post sender != address(0)
574         @post recipient != address(0)
575         @post (_balances[sender] < amount || _allowances[sender][msg.sender]
    ↪  < amount) == (__reverted)
576         @post (!__reverted && sender != recipient) ->
    ↪  (__post._balances[sender] == _balances[sender] - amount)
577         @post (!__reverted && sender != recipient) ->
    ↪  (__post._balances[recipient] == _balances[recipient] + amount)
578         @post (!__reverted && sender == recipient) ->
    ↪  (__post._balances[sender] == _balances[sender])
579         @post (!__reverted) -> (__post._allowances[sender][msg.sender] ==
    ↪  _allowances[sender][msg.sender] - amount)
580     */
581     function transferFrom(address sender, address recipient, uint256 amount)
    ↪  public virtual override returns (bool) {
582         _transfer(sender, recipient, amount);
583         _approve(sender, _msgSender(),
    ↪  _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount
    ↪  exceeds allowance"));
584         return true;
585     }
586 
587     /**
588      * @dev Atomically increases the allowance granted to `spender` by the
    ↪  caller.
589      *
590      * This is an alternative to {approve} that can be used as a mitigation
    ↪  for
591      * problems described in {IERC20-approve}.
```

```
592      *
593      * Emits an {Approval} event indicating the updated allowance.
594      *
595      * Requirements:
596      *
597      * - `spender` cannot be the zero address.
598      */
599     /*@CTK ERC20_increaseAllowance
600         @tag assume_completion
601         @post spender != address(0)
602         @post __post._allowances[msg.sender][spender] ==
    ↪ _allowances[msg.sender][spender] + addedValue
603     */
604     function increaseAllowance(address spender, uint256 addedValue) public
    ↪ virtual returns (bool) {
605         _approve(_msgSender(), spender,
    ↪ _allowances[_msgSender()][spender].add(addedValue));
606         return true;
607     }
608
609     /**
610      * @dev Atomically decreases the allowance granted to `spender` by the
    ↪ caller.
611      *
612      * This is an alternative to {approve} that can be used as a mitigation
    ↪ for
613      * problems described in {IERC20-approve}.
614      *
615      * Emits an {Approval} event indicating the updated allowance.
616      *
617      * Requirements:
618      *
619      * - `spender` cannot be the zero address.
620      * - `spender` must have allowance for the caller of at least
621      * `subtractedValue`.
622      */
623     /*@CTK ERC20_decreaseAllowance
624         @tag assume_completion
625         @post spender != address(0)
626         @post (_allowances[msg.sender][spender] < subtractedValue) ==
    ↪ __reverted
627         @post !__reverted -> __post._allowances[msg.sender][spender] ==
    ↪ _allowances[msg.sender][spender] - subtractedValue
628     */
629     function decreaseAllowance(address spender, uint256 subtractedValue)
    ↪ public virtual returns (bool) {
```

```
630         _approve(_msgSender(), spender,
    ↪ _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased
    ↪ allowance below zero"));
631         return true;
632     }
633 
634     /**
635      * @dev Moves tokens `amount` from `sender` to `recipient`.
636      *
637      * This is internal function is equivalent to {transfer}, and can be
    ↪ used to
638      * e.g. implement automatic token fees, slashing mechanisms, etc.
639      *
640      * Emits a {Transfer} event.
641      *
642      * Requirements:
643      *
644      * - `sender` cannot be the zero address.
645      * - `recipient` cannot be the zero address.
646      * - `sender` must have a balance of at least `amount`.
647      */
648     /*@CTK ERC20_transfer
649         @tag assume_completion
650         @post sender != address(0)
651         @post recipient != address(0)
652         @post (_balances[sender] < amount) == (__reverted)
653         @post (!__reverted && sender != recipient) ->
    ↪ (__post._balances[sender] == _balances[sender] - amount)
654         @post (!__reverted && sender != recipient) ->
    ↪ (__post._balances[recipient] == _balances[recipient] + amount)
655         @post (!__reverted && sender == recipient) ->
    ↪ (__post._balances[sender] == _balances[sender])
656     */
657     function _transfer(address sender, address recipient, uint256 amount)
    ↪ internal virtual {
658         require(sender != address(0), "ERC20: transfer from the zero
    ↪ address");
659         require(recipient != address(0), "ERC20: transfer to the zero
    ↪ address");
660 
661         _beforeTokenTransfer(sender, recipient, amount);
662 
663         _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer
    ↪ amount exceeds balance");
664         _balances[recipient] = _balances[recipient].add(amount);
665         emit Transfer(sender, recipient, amount);
666     }
667 
```

```
668     /** @dev Creates `amount` tokens and assigns them to `account`,
    ↪  increasing
669      * the total supply.
670      *
671      * Emits a {Transfer} event with `from` set to the zero address.
672      *
673      * Requirements
674      *
675      * - `to` cannot be the zero address.
676      */
677     /*@CTK ERC20_mint
678         @tag assume_completion
679         @post account != address(0)
680         @post __post._totalSupply == _totalSupply + amount
681         @post __post._balances[account] == _balances[account] + amount
682     */
683     function _mint(address account, uint256 amount) internal virtual {
684         require(account != address(0), "ERC20: mint to the zero address");
685 
686         _beforeTokenTransfer(address(0), account, amount);
687 
688         _totalSupply = _totalSupply.add(amount);
689         _balances[account] = _balances[account].add(amount);
690         emit Transfer(address(0), account, amount);
691     }
692 
693     /**
694      * @dev Destroys `amount` tokens from `account`, reducing the
695      * total supply.
696      *
697      * Emits a {Transfer} event with `to` set to the zero address.
698      *
699      * Requirements
700      *
701      * - `account` cannot be the zero address.
702      * - `account` must have at least `amount` tokens.
703      */
704     /*@CTK ERC20_burn
705         @tag assume_completion
706         @post account != address(0)
707         @post (_balances[account] < amount) == (__reverted)
708         @post (!__reverted) -> (__post._balances[account] ==
    ↪  _balances[account] - amount)
709         @post (!__reverted) -> (__post._totalSupply == _totalSupply -
    ↪  amount)
710     */
711     function _burn(address account, uint256 amount) internal virtual {
712         require(account != address(0), "ERC20: burn from the zero address");
```

```
713
714         _beforeTokenTransfer(account, address(0), amount);
715
716         _balances[account] = _balances[account].sub(amount, "ERC20: burn
    ↪  amount exceeds balance");
717         _totalSupply = _totalSupply.sub(amount);
718         emit Transfer(account, address(0), amount);
719     }
720
721     /**
722      * @dev Sets `amount` as the allowance of `spender` over the `owner`s
    ↪  tokens.
723      *
724      * This is internal function is equivalent to `approve`, and can be used
    ↪  to
725      * e.g. set automatic allowances for certain subsystems, etc.
726      *
727      * Emits an {Approval} event.
728      *
729      * Requirements:
730      *
731      * - `owner` cannot be the zero address.
732      * - `spender` cannot be the zero address.
733      */
734     /*@CTK ERC20__approve
735         @tag assume_completion
736         @post owner != address(0)
737         @post spender != address(0)
738         @post __post._allowances[owner][spender] == amount
739     */
740     function _approve(address owner, address spender, uint256 amount)
    ↪  internal virtual {
741         require(owner != address(0), "ERC20: approve from the zero address");
742         require(spender != address(0), "ERC20: approve to the zero address");
743
744         _allowances[owner][spender] = amount;
745         emit Approval(owner, spender, amount);
746     }
747
748     /**
749      * @dev Sets {decimals} to a value other than the default one of 18.
750      *
751      * WARNING: This function should only be called from the constructor.
    ↪  Most
752      * applications that interact with token contracts will not expect
753      * {decimals} to ever change, and may work incorrectly if it does.
754      */
755     /*@CTK ERC20__setupDecimals
```

```
756         @post __post._decimals == decimals_
757     */
758     function _setupDecimals(uint8 decimals_) internal {
759         _decimals = decimals_;
760     }
761 
762     /**
763      * @dev Hook that is called before any transfer of tokens. This includes
764      * minting and burning.
765      *
766      * Calling conditions:
767      *
768      * - when `from` and `to` are both non-zero, `amount` of ``from``'s
    ↪  tokens
769      * will be to transferred to `to`.
770      * - when `from` is zero, `amount` tokens will be minted for `to`.
771      * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
772      * - `from` and `to` are never both zero.
773      *
774      * To learn more about hooks, head to
    ↪  xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
775      */
776     function _beforeTokenTransfer(address from, address to, uint256 amount)
    ↪  internal virtual { }
777 }
778 
779 /**
780  * @dev Contract module which provides a basic access control mechanism,
    ↪  where
781  * there is an account (an owner) that can be granted exclusive access to
782  * specific functions.
783  *
784  * By default, the owner account will be the one that deploys the contract.
    ↪  This
785  * can later be changed with {transferOwnership}.
786  *
787  * This module is used through inheritance. It will make available the
    ↪  modifier
788  * `onlyOwner`, which can be applied to your functions to restrict their use
    ↪  to
789  * the owner.
790  */
791 contract Ownable is Context {
792     address private _owner;
793 
794     event OwnershipTransferred(address indexed previousOwner, address indexed
    ↪  newOwner);
795 
```

```
796     /**
797      * @dev Initializes the contract setting the deployer as the initial
    ↪ owner.
798      */
799     /*@CTK "Ownable_constructor"
800       @post __post._owner == msg.sender
801      */
802     constructor () internal {
803         address msgSender = _msgSender();
804         _owner = msgSender;
805         emit OwnershipTransferred(address(0), msgSender);
806     }
807 
808     /**
809      * @dev Returns the address of the current owner.
810      */
811     /*@CTK Ownable_owner
812         @post __return == _owner
813     */
814     function owner() public view returns (address) {
815         return _owner;
816     }
817 
818     /**
819      * @dev Throws if called by any account other than the owner.
820      */
821     modifier onlyOwner() {
822         require(_owner == _msgSender(), "Ownable: caller is not the owner");
823         _;
824     }
825 
826     /**
827      * @dev Leaves the contract without owner. It will not be possible to
    ↪ call
828      * `onlyOwner` functions anymore. Can only be called by the current
    ↪ owner.
829      *
830      * NOTE: Renouncing ownership will leave the contract without an owner,
831      * thereby removing any functionality that is only available to the
    ↪ owner.
832      */
833     /*@CTK "Ownable_renounceOwnership"
834       @tag assume_completion
835       @post _owner == msg.sender
836       @post __post._owner == address(0)
837      */
838     function renounceOwnership() public virtual onlyOwner {
839         emit OwnershipTransferred(_owner, address(0));
```

```
840         _owner = address(0);
841     }
842 
843     /**
844      * @dev Transfers ownership of the contract to a new account
    ↪  (`newOwner`).
845      * Can only be called by the current owner.
846      */
847     /*@CTK "Ownable_transferOwnership"
848       @tag assume_completion
849       @post _owner == msg.sender
850       @post newOwner != address(0)
851       @post __post._owner == newOwner
852      */
853     function transferOwnership(address newOwner) public virtual onlyOwner {
854         require(newOwner != address(0), "Ownable: new owner is the zero
    ↪  address");
855         emit OwnershipTransferred(_owner, newOwner);
856         _owner = newOwner;
857     }
858 }
859 
860 contract NFTSToken is ERC20("NFT STARS COIN", "NFTS"), Ownable {
861 
862     constructor() public {
863         _mint(owner(), 200000000000000000000000000);
864     }
865 
866 }
```