CertiK Audit Report For 12Ships

Request Date: 2019-05-09
Revision Date: 2019-05-13
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 12Ships(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.

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by 12Ships. This audit was conducted to discover issues and vulnerabilities in the source code of 12Ships's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice 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

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

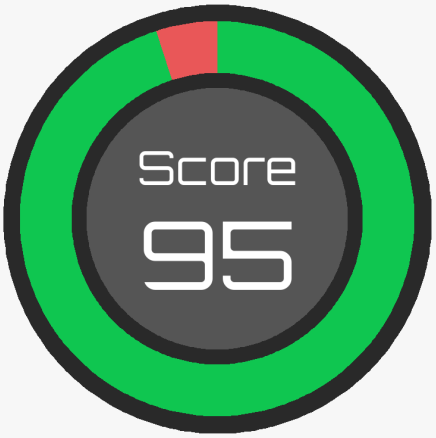
- Critical: The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.
- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

May 13, 2019



Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to 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 minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an 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.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors 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
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, 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. A malicious user is able to 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 as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

In function `_unlock()`, `timelockList[holder].length -=1;` has the possibility to overflow. However, this is considered low risk. Whether or not to fix it is up to the author.

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     }
56
```

Post environment

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

Formal Verification Request 1

SafeMath mul

📅 13, May 2019
⏱ 340.57 ms

Line 7-13 in File 12ShipsToken.sol

```
7   /*@CTK "SafeMath mul"
8     @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
9     @post !__reverted -> __return == a * b
10    @post !__reverted == !__has_overflow
11    @post !(__has_buf_overflow)
12    @post !(__has_assertion_failure)
13   */
```

Line 14-26 in File 12ShipsToken.sol

```
14  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
15      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
16      // benefit is lost if 'b' is also tested.
17      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
18      if (a == 0) {
19          return 0;
20      }
21
22      uint256 c = a * b;
23      require(c / a == b);
24
25      return c;
26  }
```

✅ The code meets the specification

Formal Verification Request 2

SafeMath div

📅 13, May 2019
⏱ 15.79 ms

Line 31-37 in File 12ShipsToken.sol

```
31  /*@CTK "SafeMath div"
32    @post b != 0 -> !__reverted
33    @post !__reverted -> __return == a / b
34    @post !__reverted -> !__has_overflow
35    @post !(__has_buf_overflow)
36    @post !(__has_assertion_failure)
37  */
```

Line 38-45 in File 12ShipsToken.sol

```
38  function div(uint256 a, uint256 b) internal pure returns (uint256) {
39      // Solidity only automatically asserts when dividing by 0
40      require(b > 0);
41      uint256 c = a / b;
42      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```

```
43
44          return c;
45      }
```

The code meets the specification

Formal Verification Request 3

SafeMath sub

13, May 2019
14.62 ms

Line 50-56 in File 12ShipsToken.sol

```
50      /*@CTK "SafeMath sub"
51        @post (a < b) == __reverted
52        @post !__reverted -> __return == a - b
53        @post !__reverted -> !__has_overflow
54        @post !(__has_buf_overflow)
55        @post !(__has_assertion_failure)
56      */
```

Line 57-62 in File 12ShipsToken.sol

```
57      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
58          require(b <= a);
59          uint256 c = a - b;
60
61          return c;
62      }
```

The code meets the specification

Formal Verification Request 4

SafeMath add

13, May 2019
17.63 ms

Line 67-73 in File 12ShipsToken.sol

```
67      /*@CTK "SafeMath add"
68        @post (a + b < a || a + b < b) == __reverted
69        @post !__reverted -> __return == a + b
70        @post !__reverted -> !__has_overflow
71        @post !(__has_buf_overflow)
72        @post !(__has_assertion_failure)
73      */
```

Line 74-79 in File 12ShipsToken.sol

```
74      function add(uint256 a, uint256 b) internal pure returns (uint256) {
75          uint256 c = a + b;
76          require(c >= a);
77
```

```
78          return c;
79      }
```

The code meets the specification

Formal Verification Request 5

SafeMath div

13, May 2019
14.1 ms

Line 85-91 in File 12ShipsToken.sol

```
85      /*@CTK "SafeMath div"
86        @post b != 0 -> !__reverted
87        @post !__reverted -> __return == a % b
88        @post !__reverted -> !__has_overflow
89        @post !(__has_buf_overflow)
90        @post !(__has_assertion_failure)
91       */
```

Line 92-95 in File 12ShipsToken.sol

```
92      function mod(uint256 a, uint256 b) internal pure returns (uint256) {
93          require(b != 0);
94          return a % b;
95      }
```

The code meets the specification

Formal Verification Request 6

If method completes, integer overflow would not happen.

13, May 2019
20.36 ms

Line 106 in File 12ShipsToken.sol

```
106     //@CTK NO_OVERFLOW
```

Line 114-119 in File 12ShipsToken.sol

```
114     function add(Role storage role, address account) internal {
115         require(account != address(0));
116         require(!role.bearer[account]);
117
118         role.bearer[account] = true;
119     }
```

The code meets the specification

Formal Verification Request 7

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

13, May 2019
0.6 ms

Line 107 in File 12ShipsToken.sol

```
107     //@CTK NO_BUF_OVERFLOW
```

Line 114-119 in File 12ShipsToken.sol

```
114     function add(Role storage role, address account) internal {
115         require(account != address(0));
116         require(!role.bearer[account]);
117 
118         role.bearer[account] = true;
119     }
```

The code meets the specification

Formal Verification Request 8

Method will not encounter an assertion failure.

13, May 2019
0.63 ms

Line 108 in File 12ShipsToken.sol

```
108     //@CTK NO_ASF
```

Line 114-119 in File 12ShipsToken.sol

```
114     function add(Role storage role, address account) internal {
115         require(account != address(0));
116         require(!role.bearer[account]);
117 
118         role.bearer[account] = true;
119     }
```

The code meets the specification

Formal Verification Request 9

Roles add correctness

13, May 2019
2.11 ms

Line 109-113 in File 12ShipsToken.sol

```
109     /*@CTK "Roles add correctness"
110       @post account == 0x0 -> __reverted
111       @post role.bearer[account] -> __reverted
112       @post account != 0x0 && !role.bearer[account] -> !__reverted
113      */
```

Line 114-119 in File 12ShipsToken.sol

```
114     function add(Role storage role, address account) internal {
115         require(account != address(0));
116         require(!role.bearer[account]);
117 
118         role.bearer[account] = true;
119     }
```

✅ The code meets the specification

Formal Verification Request 10

If method completes, integer overflow would not happen.

📅 13, May 2019
⏱ 19.91 ms

Line 124 in File 12ShipsToken.sol

```
124     //@CTK NO_OVERFLOW
```

Line 132-137 in File 12ShipsToken.sol

```
132     function remove(Role storage role, address account) internal {
133         require(account != address(0));
134         require(role.bearer[account]);
135 
136         role.bearer[account] = false;
137     }
```

✅ The code meets the specification

Formal Verification Request 11

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

📅 13, May 2019
⏱ 0.53 ms

Line 125 in File 12ShipsToken.sol

```
125     //@CTK NO_BUF_OVERFLOW
```

Line 132-137 in File 12ShipsToken.sol

```
132     function remove(Role storage role, address account) internal {
133         require(account != address(0));
134         require(role.bearer[account]);
135 
136         role.bearer[account] = false;
137     }
```

✅ The code meets the specification

Formal Verification Request 12

Method will not encounter an assertion failure.

13, May 2019
0.52 ms

Line 126 in File 12ShipsToken.sol

```
126     //@CTK NO_ASF
```

Line 132-137 in File 12ShipsToken.sol

```
132     function remove(Role storage role, address account) internal {
133         require(account != address(0));
134         require(role.bearer[account]);
135 
136         role.bearer[account] = false;
137     }
```

The code meets the specification

Formal Verification Request 13

Roles add correctness

13, May 2019
1.84 ms

Line 127-131 in File 12ShipsToken.sol

```
127     /*@CTK "Roles add correctness"
128       @post account == 0x0 -> __reverted
129       @post !role.bearer[account] -> __reverted
130       @post account != 0x0 && role.bearer[account] -> !__reverted
131      */
```

Line 132-137 in File 12ShipsToken.sol

```
132     function remove(Role storage role, address account) internal {
133         require(account != address(0));
134         require(role.bearer[account]);
135 
136         role.bearer[account] = false;
137     }
```

The code meets the specification

Formal Verification Request 14

If method completes, integer overflow would not happen.

13, May 2019
14.21 ms

Line 143 in File 12ShipsToken.sol

```
143     //@CTK NO_OVERFLOW
```

Line 150-153 in File 12ShipsToken.sol

```
150     function has(Role storage role, address account) internal view returns (bool) {
151         require(account != address(0));
152         return role.bearer[account];
153     }
```

The code meets the specification

Formal Verification Request 15

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

13, May 2019
0.57 ms

Line 144 in File 12ShipsToken.sol

```
144     //@CTK NO_BUF_OVERFLOW
```

Line 150-153 in File 12ShipsToken.sol

```
150     function has(Role storage role, address account) internal view returns (bool) {
151         require(account != address(0));
152         return role.bearer[account];
153     }
```

The code meets the specification

Formal Verification Request 16

Method will not encounter an assertion failure.

13, May 2019
0.58 ms

Line 145 in File 12ShipsToken.sol

```
145     //@CTK NO_ASF
```

Line 150-153 in File 12ShipsToken.sol

```
150     function has(Role storage role, address account) internal view returns (bool) {
151         require(account != address(0));
152         return role.bearer[account];
153     }
```

The code meets the specification

Formal Verification Request 17

Roles has correctness

13, May 2019
1.23 ms

Line 146-149 in File 12ShipsToken.sol

```
146     /*@CTK "Roles has correctness"
147       @post account == 0x0 -> __reverted
148       @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
149      */
```

Line 150-153 in File 12ShipsToken.sol

```
150     function has(Role storage role, address account) internal view returns (bool) {
151         require(account != address(0));
152         return role.bearer[account];
153     }
```

The code meets the specification

Formal Verification Request 18

If method completes, integer overflow would not happen.

13, May 2019
130.98 ms

Line 206 in File 12ShipsToken.sol

```
206     //@CTK NO_OVERFLOW
```

Line 215-217 in File 12ShipsToken.sol

```
215     constructor () internal {
216         _addPauser(msg.sender);
217     }
```

The code meets the specification

Formal Verification Request 19

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

13, May 2019
0.8 ms

Line 207 in File 12ShipsToken.sol

```
207     //@CTK NO_BUF_OVERFLOW
```

Line 215-217 in File 12ShipsToken.sol

```
215     constructor () internal {
216         _addPauser(msg.sender);
217     }
```

The code meets the specification

Formal Verification Request 20

Method will not encounter an assertion failure.

13, May 2019
0.78 ms

Line 208 in File 12ShipsToken.sol

```
208     //@CTK NO_ASF
```

Line 215-217 in File 12ShipsToken.sol

```
215     constructor () internal {
216         _addPauser(msg.sender);
217     }
```

The code meets the specification

Formal Verification Request 21

PauserRole constructor correctness

13, May 2019
3.86 ms

Line 209-214 in File 12ShipsToken.sol

```
209     /*@CTK "PauserRole constructor correctness"
210       @post msg.sender == 0x0 -> __reverted
211       @post _pausers.bearer[msg.sender] -> __reverted
212       @post msg.sender != 0x0 && !_pausers.bearer[msg.sender]
213           -> !__reverted && __post._pausers.bearer[msg.sender]
214      */
```

Line 215-217 in File 12ShipsToken.sol

```
215     constructor () internal {
216         _addPauser(msg.sender);
217     }
```

The code meets the specification

Formal Verification Request 22

If method completes, integer overflow would not happen.

13, May 2019
32.69 ms

Line 224 in File 12ShipsToken.sol

```
224     //@CTK NO_OVERFLOW
```

Line 231-233 in File 12ShipsToken.sol

```
231     function isPauser(address account) public view returns (bool) {
232         return _pausers.has(account);
233     }
```

The code meets the specification

Formal Verification Request 23

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

13, May 2019
0.72 ms

Line 225 in File 12ShipsToken.sol

```
225     //@CTK NO_BUF_OVERFLOW
```

Line 231-233 in File 12ShipsToken.sol

```
231     function isPauser(address account) public view returns (bool) {
232         return _pausers.has(account);
233     }
```

The code meets the specification

Formal Verification Request 24

Method will not encounter an assertion failure.

13, May 2019
0.68 ms

Line 226 in File 12ShipsToken.sol

```
226     //@CTK NO_ASF
```

Line 231-233 in File 12ShipsToken.sol

```
231     function isPauser(address account) public view returns (bool) {
232         return _pausers.has(account);
233     }
```

The code meets the specification

Formal Verification Request 25

isBurner correctness

13, May 2019
1.9 ms

Line 227-230 in File 12ShipsToken.sol

```
227     /*@CTK "isBurner correctness"
228       @post account == 0x0 -> __reverted
229       @post account != 0x0 -> !__reverted && __return == _pausers.bearer[account]
230      */
```

Line 231-233 in File 12ShipsToken.sol

```
231     function isPauser(address account) public view returns (bool) {
232         return _pausers.has(account);
233     }
```

The code meets the specification

Formal Verification Request 26

If method completes, integer overflow would not happen.

13, May 2019
132.62 ms

Line 235 in File 12ShipsToken.sol

```
235     //@CTK NO_OVERFLOW
```

Line 250-252 in File 12ShipsToken.sol

```
250     function addPauser(address account) public onlyPauser {
251         _addPauser(account);
252     }
```

The code meets the specification

Formal Verification Request 27

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

13, May 2019
5.53 ms

Line 236 in File 12ShipsToken.sol

```
236     //@CTK NO_BUF_OVERFLOW
```

Line 250-252 in File 12ShipsToken.sol

```
250     function addPauser(address account) public onlyPauser {
251         _addPauser(account);
252     }
```

The code meets the specification

Formal Verification Request 28

Method will not encounter an assertion failure.

13, May 2019
5.33 ms

Line 237 in File 12ShipsToken.sol

```
237     //@CTK NO_ASF
```

Line 250-252 in File 12ShipsToken.sol

```
250     function addPauser(address account) public onlyPauser {
251         _addPauser(account);
252     }
```

The code meets the specification

Formal Verification Request 29

addPauser correctness

13, May 2019
8.04 ms

Line 238-249 in File 12ShipsToken.sol

```
238     /*@CTK "addPauser correctness"
239       @post account == 0x0 -> __reverted
240       @post msg.sender == 0x0 -> __reverted
241       @post _pausers.bearer[account] -> __reverted
242       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
243       @post account != 0x0 && !_pausers.bearer[account]
244           && msg.sender != 0x0 && _pausers.bearer[msg.sender]
245           -> !__reverted && __post._pausers.bearer[account]
246       @post account != 0x0 && !_pausers.bearer[account]
247           && msg.sender != 0x0 && owner == msg.sender
248           -> !__reverted && __post._pausers.bearer[account]
249      */
```

Line 250-252 in File 12ShipsToken.sol

```
250     function addPauser(address account) public onlyPauser {
251         _addPauser(account);
252     }
```

The code meets the specification

Formal Verification Request 30

If method completes, integer overflow would not happen.

13, May 2019
87.73 ms

Line 254 in File 12ShipsToken.sol

```
254     //@CTK NO_OVERFLOW
```

Line 264-266 in File 12ShipsToken.sol

```
264     function removePauser(address account) public onlyOwner {
265         _removePauser(account);
266     }
```

The code meets the specification

Formal Verification Request 31

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

13, May 2019
1.07 ms

Line 255 in File 12ShipsToken.sol

```
255     //@CTK NO_BUF_OVERFLOW
```

Line 264-266 in File 12ShipsToken.sol

```
264     function removePauser(address account) public onlyOwner {
265         _removePauser(account);
266     }
```

The code meets the specification

Formal Verification Request 32

Method will not encounter an assertion failure.

13, May 2019
1.04 ms

Line 256 in File 12ShipsToken.sol

```
256     //@CTK NO_ASF
```

Line 264-266 in File 12ShipsToken.sol

```
264     function removePauser(address account) public onlyOwner {
265         _removePauser(account);
266     }
```

The code meets the specification

Formal Verification Request 33

removePauser correctness

13, May 2019
5.73 ms

Line 257-263 in File 12ShipsToken.sol

```
257     /*@CTK "removePauser correctness"
258       @post account == 0x0 -> __reverted
259       @post !_pausers.bearer[account] -> __reverted
260       @post owner != msg.sender -> __reverted
261       @post account != 0x0 && _pausers.bearer[account] && owner == msg.sender
262           -> !__reverted && !__post._pausers.bearer[account]
263      */
```

Line 264-266 in File 12ShipsToken.sol

```
264     function removePauser(address account) public onlyOwner {
265         _removePauser(account);
266     }
```

The code meets the specification

Formal Verification Request 34

If method completes, integer overflow would not happen.

13, May 2019
48.53 ms

Line 268 in File 12ShipsToken.sol

```
268     //@CTK NO_OVERFLOW
```

Line 277-279 in File 12ShipsToken.sol

```
277     function renouncePauser() public {
278         _removePauser(msg.sender);
279     }
```

The code meets the specification

Formal Verification Request 35

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

13, May 2019
0.79 ms

Line 269 in File 12ShipsToken.sol

```
269     //@CTK NO_BUF_OVERFLOW
```

Line 277-279 in File 12ShipsToken.sol

```
277     function renouncePauser() public {
278         _removePauser(msg.sender);
279     }
```

The code meets the specification

Formal Verification Request 36

Method will not encounter an assertion failure.

13, May 2019
0.76 ms

Line 270 in File 12ShipsToken.sol

```
270     //@CTK NO_ASF
```

Line 277-279 in File 12ShipsToken.sol

```
277     function renouncePauser() public {
278         _removePauser(msg.sender);
279     }
```

The code meets the specification

Formal Verification Request 37

renouncePauser correctness

13, May 2019
3.75 ms

Line 271-276 in File 12ShipsToken.sol

```
271     /*@CTK "renouncePauser correctness"
272       @post msg.sender == 0x0 -> __reverted
273       @post !_pausers.bearer[msg.sender] -> __reverted
274       @post msg.sender != 0x0 && _pausers.bearer[msg.sender]
275           -> !__reverted && !__post._pausers.bearer[msg.sender]
276      */
```

Line 277-279 in File 12ShipsToken.sol

```
277     function renouncePauser() public {
278         _removePauser(msg.sender);
279     }
```

The code meets the specification

Formal Verification Request 38

If method completes, integer overflow would not happen.

13, May 2019
5.37 ms

Line 298 in File 12ShipsToken.sol

```
298     //@CTK NO_OVERFLOW
```

Line 304-306 in File 12ShipsToken.sol

```
304     constructor () internal {
305         _paused = false;
306     }
```

The code meets the specification

Formal Verification Request 39

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

13, May 2019
0.41 ms

Line 299 in File 12ShipsToken.sol

```
299     //@CTK NO_BUF_OVERFLOW
```

Line 304-306 in File 12ShipsToken.sol

```
304     constructor () internal {
305         _paused = false;
306     }
```

✅ The code meets the specification

Formal Verification Request 40

Method will not encounter an assertion failure.

📅 13, May 2019
⏱ 0.41 ms

Line 300 in File 12ShipsToken.sol

```
300     //@CTK NO_ASF
```

Line 304-306 in File 12ShipsToken.sol

```
304     constructor () internal {
305         _paused = false;
306     }
```

✅ The code meets the specification

Formal Verification Request 41

Pausable constructor correctness

📅 13, May 2019
⏱ 0.91 ms

Line 301-303 in File 12ShipsToken.sol

```
301     /*@CTK "Pausable constructor correctness"
302       @post __post._paused == false
303      */
```

Line 304-306 in File 12ShipsToken.sol

```
304     constructor () internal {
305         _paused = false;
306     }
```

✅ The code meets the specification

Formal Verification Request 42

If method completes, integer overflow would not happen.

📅 13, May 2019
⏱ 5.71 ms

Line 311 in File 12ShipsToken.sol

```
311     //@CTK NO_OVERFLOW
```

Line 317-319 in File 12ShipsToken.sol

```
317     function paused() public view returns (bool) {
318         return _paused;
319     }
```

✅ The code meets the specification

Formal Verification Request 43

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

📅 13, May 2019
⏱ 0.4 ms

Line 312 in File 12ShipsToken.sol

```
312     //@CTK NO_BUF_OVERFLOW
```

Line 317-319 in File 12ShipsToken.sol

```
317     function paused() public view returns (bool) {
318         return _paused;
319     }
```

✅ The code meets the specification

Formal Verification Request 44

Method will not encounter an assertion failure.

📅 13, May 2019
⏱ 0.38 ms

Line 313 in File 12ShipsToken.sol

```
313     //@CTK NO_ASF
```

Line 317-319 in File 12ShipsToken.sol

```
317     function paused() public view returns (bool) {
318         return _paused;
319     }
```

✅ The code meets the specification

Formal Verification Request 45

Pausable paused correctness

📅 13, May 2019
⏱ 0.47 ms

Line 314-316 in File 12ShipsToken.sol

```
314     /*@CTK "Pausable paused correctness"
315       @post __return == _paused
316      */
```

Line 317-319 in File 12ShipsToken.sol

```
317     function paused() public view returns (bool) {
318         return _paused;
319     }
```

The code meets the specification

Formal Verification Request 46

If method completes, integer overflow would not happen.

13, May 2019
128.0 ms

Line 340 in File 12ShipsToken.sol

```
340     //@CTK NO_OVERFLOW
```

Line 352-355 in File 12ShipsToken.sol

```
352     function pause() public onlyPauser whenNotPaused {
353         _paused = true;
354         emit Paused(msg.sender);
355     }
```

The code meets the specification

Formal Verification Request 47

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

13, May 2019
1.36 ms

Line 341 in File 12ShipsToken.sol

```
341     //@CTK NO_BUF_OVERFLOW
```

Line 352-355 in File 12ShipsToken.sol

```
352     function pause() public onlyPauser whenNotPaused {
353         _paused = true;
354         emit Paused(msg.sender);
355     }
```

The code meets the specification

Formal Verification Request 48

Method will not encounter an assertion failure.

13, May 2019
1.3 ms

Line 342 in File 12ShipsToken.sol

```
342     //@CTK NO_ASF
```

Line 352-355 in File 12ShipsToken.sol

```
352     function pause() public onlyPauser whenNotPaused {
353         _paused = true;
354         emit Paused(msg.sender);
355     }
```

The code meets the specification

Formal Verification Request 49

Pausable pause correctness

13, May 2019
31.37 ms

Line 343-351 in File 12ShipsToken.sol

```
343     /*@CTK "Pausable pause correctness"
344       @post _paused -> __reverted
345       @post msg.sender == 0x0 -> __reverted
346       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
347       @post !_paused && msg.sender != 0x0 && _pausers.bearer[msg.sender]
348           -> __post._paused
349       @post !_paused && msg.sender != 0x0 && owner == msg.sender
350           -> __post._paused
351     */
```

Line 352-355 in File 12ShipsToken.sol

```
352     function pause() public onlyPauser whenNotPaused {
353         _paused = true;
354         emit Paused(msg.sender);
355     }
```

The code meets the specification

Formal Verification Request 50

If method completes, integer overflow would not happen.

13, May 2019
83.61 ms

Line 360 in File 12ShipsToken.sol

```
360     //@CTK NO_OVERFLOW
```

Line 372-375 in File 12ShipsToken.sol

```
372     function unpause() public onlyPauser whenPaused {
373         _paused = false;
374         emit Unpaused(msg.sender);
375     }
```

The code meets the specification

Formal Verification Request 51

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

13, May 2019
1.38 ms

Line 361 in File 12ShipsToken.sol

```
361     //@CTK NO_BUF_OVERFLOW
```

Line 372-375 in File 12ShipsToken.sol

```
372     function unpause() public onlyPauser whenPaused {
373         _paused = false;
374         emit Unpaused(msg.sender);
375     }
```

The code meets the specification

Formal Verification Request 52

Method will not encounter an assertion failure.

13, May 2019
1.32 ms

Line 362 in File 12ShipsToken.sol

```
362     //@CTK NO_ASF
```

Line 372-375 in File 12ShipsToken.sol

```
372     function unpause() public onlyPauser whenPaused {
373         _paused = false;
374         emit Unpaused(msg.sender);
375     }
```

The code meets the specification

Formal Verification Request 53

Pausable unpause correctness

13, May 2019
8.29 ms

Line 363-371 in File 12ShipsToken.sol

```
363     /*@CTK "Pausable unpause correctness"
364       @post !_paused -> __reverted
365       @post msg.sender == 0x0 -> __reverted
366       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
367       @post _paused && msg.sender != 0x0 && _pausers.bearer[msg.sender]
368           -> !__post._paused
369       @post _paused && msg.sender != 0x0 && owner == msg.sender
370           -> !__post._paused
371     */
```

Line 372-375 in File 12ShipsToken.sol

```
372     function unpause() public onlyPauser whenPaused {
373         _paused = false;
374         emit Unpaused(msg.sender);
375     }
```

The code meets the specification

Formal Verification Request 54

If method completes, integer overflow would not happen.

13, May 2019
5.44 ms

Line 408 in File 12ShipsToken.sol

```
408     //@CTK NO_OVERFLOW
```

Line 414-416 in File 12ShipsToken.sol

```
414     function totalSupply() public view returns (uint256) {
415         return _totalSupply;
416     }
```

The code meets the specification

Formal Verification Request 55

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

13, May 2019
0.38 ms

Line 409 in File 12ShipsToken.sol

```
409     //@CTK NO_BUF_OVERFLOW
```

Line 414-416 in File 12ShipsToken.sol

```
414     function totalSupply() public view returns (uint256) {
415         return _totalSupply;
416     }
```

✅ The code meets the specification

Formal Verification Request 56

Method will not encounter an assertion failure.

📅 13, May 2019
⏱ 0.42 ms

Line 410 in File 12ShipsToken.sol

```
410     //@CTK NO_ASF
```

Line 414-416 in File 12ShipsToken.sol

```
414     function totalSupply() public view returns (uint256) {
415         return _totalSupply;
416     }
```

✅ The code meets the specification

Formal Verification Request 57

totalSupply correctness

📅 13, May 2019
⏱ 0.42 ms

Line 411-413 in File 12ShipsToken.sol

```
411     /*@CTK "totalSupply correctness"
412       @post __return == _totalSupply
413      */
```

Line 414-416 in File 12ShipsToken.sol

```
414     function totalSupply() public view returns (uint256) {
415         return _totalSupply;
416     }
```

✅ The code meets the specification

Formal Verification Request 58

If method completes, integer overflow would not happen.

📅 13, May 2019
⏱ 5.65 ms

Line 423 in File 12ShipsToken.sol

```
423     //@CTK NO_OVERFLOW
```

Line 429-431 in File 12ShipsToken.sol

```
429     function balanceOf(address owner) public view returns (uint256) {
430         return _balances[owner];
431     }
```

The code meets the specification

Formal Verification Request 59

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

13, May 2019
0.4 ms

Line 424 in File 12ShipsToken.sol

```
424     //@CTK NO_BUF_OVERFLOW
```

Line 429-431 in File 12ShipsToken.sol

```
429     function balanceOf(address owner) public view returns (uint256) {
430         return _balances[owner];
431     }
```

The code meets the specification

Formal Verification Request 60

Method will not encounter an assertion failure.

13, May 2019
0.38 ms

Line 425 in File 12ShipsToken.sol

```
425     //@CTK NO_ASF
```

Line 429-431 in File 12ShipsToken.sol

```
429     function balanceOf(address owner) public view returns (uint256) {
430         return _balances[owner];
431     }
```

The code meets the specification

Formal Verification Request 61

balanceOf correctness

13, May 2019
0.43 ms

Line 426-428 in File 12ShipsToken.sol

```
426     /*@CTK "balanceOf correctness"
427       @post __return == _balances[owner]
428      */
```

Line 429-431 in File 12ShipsToken.sol

```
429     function balanceOf(address owner) public view returns (uint256) {
430         return _balances[owner];
431     }
```

The code meets the specification

Formal Verification Request 62

If method completes, integer overflow would not happen.

13, May 2019
5.8 ms

Line 439 in File 12ShipsToken.sol

```
439     //@CTK NO_OVERFLOW
```

Line 445-447 in File 12ShipsToken.sol

```
445     function allowance(address owner, address spender) public view returns (uint256) {
446         return _allowed[owner][spender];
447     }
```

The code meets the specification

Formal Verification Request 63

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

13, May 2019
0.62 ms

Line 440 in File 12ShipsToken.sol

```
440     //@CTK NO_BUF_OVERFLOW
```

Line 445-447 in File 12ShipsToken.sol

```
445     function allowance(address owner, address spender) public view returns (uint256) {
446         return _allowed[owner][spender];
447     }
```

The code meets the specification

Formal Verification Request 64

Method will not encounter an assertion failure.

13, May 2019
0.61 ms

Line 441 in File 12ShipsToken.sol

```
441     //@CTK NO_ASF
```

Line 445-447 in File 12ShipsToken.sol

```
445     function allowance(address owner, address spender) public view returns (uint256) {
446         return _allowed[owner][spender];
447     }
```

The code meets the specification

Formal Verification Request 65

allowance correctness

13, May 2019
0.48 ms

Line 442-444 in File 12ShipsToken.sol

```
442     /*@CTK "allowance correctness"
443       @post __return == _allowed[owner][spender]
444      */
```

Line 445-447 in File 12ShipsToken.sol

```
445     function allowance(address owner, address spender) public view returns (uint256) {
446         return _allowed[owner][spender];
447     }
```

The code meets the specification

Formal Verification Request 66

If method completes, integer overflow would not happen.

13, May 2019
162.93 ms

Line 454 in File 12ShipsToken.sol

```
454     //@CTK NO_OVERFLOW
```

Line 465-468 in File 12ShipsToken.sol

```
465     function transfer(address to, uint256 value) public returns (bool) {
466         _transfer(msg.sender, to, value);
467         return true;
468     }
```

The code meets the specification

Formal Verification Request 67

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

13, May 2019
9.42 ms

Line 455 in File 12ShipsToken.sol

```
455     //@CTK NO_BUF_OVERFLOW
```

Line 465-468 in File 12ShipsToken.sol

```
465     function transfer(address to, uint256 value) public returns (bool) {
466         _transfer(msg.sender, to, value);
467         return true;
468     }
```

The code meets the specification

Formal Verification Request 68

Method will not encounter an assertion failure.

13, May 2019
10.06 ms

Line 456 in File 12ShipsToken.sol

```
456     //@CTK NO_ASF
```

Line 465-468 in File 12ShipsToken.sol

```
465     function transfer(address to, uint256 value) public returns (bool) {
466         _transfer(msg.sender, to, value);
467         return true;
468     }
```

The code meets the specification

Formal Verification Request 69

transfer correctness

13, May 2019
90.44 ms

Line 457-464 in File 12ShipsToken.sol

```
457     /*@CTK "transfer correctness"
458       @tag assume_completion
459       @post to != 0x0
460       @post value <= _balances[msg.sender]
461       @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - value
462       @post to != msg.sender -> __post._balances[to] == _balances[to] + value
463       @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
464      */
```

Line 465-468 in File 12ShipsToken.sol

```
465     function transfer(address to, uint256 value) public returns (bool) {
466         _transfer(msg.sender, to, value);
467         return true;
468     }
```

The code meets the specification

Formal Verification Request 70

If method completes, integer overflow would not happen.

13, May 2019
16.96 ms

Line 479 in File 12ShipsToken.sol

```
479     //@CTK NO_OVERFLOW
```

Line 486-492 in File 12ShipsToken.sol

```
486     function approve(address spender, uint256 value) public returns (bool) {
487         require(spender != address(0));
488 
489         _allowed[msg.sender][spender] = value;
490         emit Approval(msg.sender, spender, value);
491         return true;
492     }
```

The code meets the specification

Formal Verification Request 71

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

13, May 2019
0.65 ms

Line 480 in File 12ShipsToken.sol

```
480     //@CTK NO_BUF_OVERFLOW
```

Line 486-492 in File 12ShipsToken.sol

```
486     function approve(address spender, uint256 value) public returns (bool) {
487         require(spender != address(0));
488 
489         _allowed[msg.sender][spender] = value;
490         emit Approval(msg.sender, spender, value);
491         return true;
492     }
```

The code meets the specification

Formal Verification Request 72

Method will not encounter an assertion failure.

13, May 2019
0.52 ms

Line 481 in File 12ShipsToken.sol

```
481     //@CTK NO_ASF
```

Line 486-492 in File 12ShipsToken.sol

```
486     function approve(address spender, uint256 value) public returns (bool) {
487         require(spender != address(0));
488 
489         _allowed[msg.sender][spender] = value;
490         emit Approval(msg.sender, spender, value);
491         return true;
492     }
```

The code meets the specification

Formal Verification Request 73

approve correctness

13, May 2019
1.8 ms

Line 482-485 in File 12ShipsToken.sol

```
482     /*@CTK "approve correctness"
483       @post spender == 0x0 -> __reverted
484       @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
485      */
```

Line 486-492 in File 12ShipsToken.sol

```
486     function approve(address spender, uint256 value) public returns (bool) {
487         require(spender != address(0));
488 
489         _allowed[msg.sender][spender] = value;
490         emit Approval(msg.sender, spender, value);
491         return true;
492     }
```

The code meets the specification

Formal Verification Request 74

If method completes, integer overflow would not happen.

13, May 2019
116.01 ms

Line 502 in File 12ShipsToken.sol

```
502     //@CTK NO_OVERFLOW
```

Line 514-519 in File 12ShipsToken.sol

```
514     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
515         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
516         _transfer(from, to, value);
517         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
518         return true;
519     }
```

The code meets the specification

Formal Verification Request 75

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

13, May 2019
10.65 ms

Line 503 in File 12ShipsToken.sol

```
503     //@CTK NO_BUF_OVERFLOW
```

Line 514-519 in File 12ShipsToken.sol

```
514     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
515         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
516         _transfer(from, to, value);
517         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
518         return true;
519     }
```

The code meets the specification

Formal Verification Request 76

Method will not encounter an assertion failure.

13, May 2019
9.83 ms

Line 504 in File 12ShipsToken.sol

```
504     //@CTK NO_ASF
```

Line 514-519 in File 12ShipsToken.sol

```
514     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
515         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
516         _transfer(from, to, value);
517         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
518         return true;
519     }
```

The code meets the specification

Formal Verification Request 77

transferFrom correctness

13, May 2019
179.9 ms

Line 505-513 in File 12ShipsToken.sol

```
505     /*@CTK "transferFrom correctness"
506       @tag assume_completion
507       @post to != 0x0
508       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
509       @post to != from -> __post._balances[from] == _balances[from] - value
510       @post to != from -> __post._balances[to] == _balances[to] + value
511       @post to == from -> __post._balances[from] == _balances[from]
512       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
513      */
```

Line 514-519 in File 12ShipsToken.sol

```
514     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
515         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
516         _transfer(from, to, value);
517         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
518         return true;
519     }
```

The code meets the specification

Formal Verification Request 78

If method completes, integer overflow would not happen.

13, May 2019
43.02 ms

Line 531 in File 12ShipsToken.sol

```
531     //@CTK NO_OVERFLOW
```

Line 539-545 in File 12ShipsToken.sol

```
539     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
540         require(spender != address(0));
541 
542         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
543         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
544         return true;
545     }
```

The code meets the specification

Formal Verification Request 79

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

13, May 2019
0.92 ms

Line 532 in File 12ShipsToken.sol

```
532     //@CTK NO_BUF_OVERFLOW
```

Line 539-545 in File 12ShipsToken.sol

```
539     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
540         require(spender != address(0));
541 
542         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
543         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
544         return true;
545     }
```

The code meets the specification

Formal Verification Request 80

Method will not encounter an assertion failure.

13, May 2019
0.95 ms

Line 533 in File 12ShipsToken.sol

```
533     //@CTK NO_ASF
```

Line 539-545 in File 12ShipsToken.sol

```
539     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
540         require(spender != address(0));
541 
542         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
543         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
544         return true;
545     }
```

The code meets the specification

Formal Verification Request 81

increaseAllowance correctness

13, May 2019
11.99 ms

Line 534-538 in File 12ShipsToken.sol

```
534     /*@CTK "increaseAllowance correctness"
535       @tag assume_completion
536       @post spender != 0x0
537       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
              addedValue
538      */
```

Line 539-545 in File 12ShipsToken.sol

```
539     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
540         require(spender != address(0));
541 
542         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
543         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
544         return true;
545     }
```

The code meets the specification

Formal Verification Request 82

If method completes, integer overflow would not happen.

13, May 2019
40.89 ms

Line 557 in File 12ShipsToken.sol

```
557     //@CTK NO_OVERFLOW
```

Line 565-571 in File 12ShipsToken.sol

```
565     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
566         require(spender != address(0));
567 
568         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
569         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
570         return true;
571     }
```

The code meets the specification

Formal Verification Request 83

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

13, May 2019
0.92 ms

Line 558 in File 12ShipsToken.sol

```
558     //@CTK NO_BUF_OVERFLOW
```

Line 565-571 in File 12ShipsToken.sol

```
565     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
566         require(spender != address(0));
567 
568         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
569         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
570         return true;
571     }
```

The code meets the specification

Formal Verification Request 84

Method will not encounter an assertion failure.

13, May 2019
0.91 ms

Line 559 in File 12ShipsToken.sol

```
559     //@CTK NO_ASF
```

Line 565-571 in File 12ShipsToken.sol

```
565     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
566         require(spender != address(0));
567 
568         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
569         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
570         return true;
571     }
```

The code meets the specification

Formal Verification Request 85

decreaseAllowance correctness

13, May 2019
16.91 ms

Line 560-564 in File 12ShipsToken.sol

```
560     /*@CTK "decreaseAllowance correctness"
561       @tag assume_completion
562       @post spender != 0x0
563       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
564      */
```

Line 565-571 in File 12ShipsToken.sol

```
565     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
566         require(spender != address(0));
567
568         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
569         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
570         return true;
571     }
```

The code meets the specification

Formal Verification Request 86

ERC20Detailed constructor correctness

13, May 2019
9.64 ms

Line 666-670 in File 12ShipsToken.sol

```
666     /*@CTK "ERC20Detailed constructor correctness"
667       @post __post._name == name
668       @post __post._symbol == symbol
669       @post __post._decimals == decimals
670      */
```

Line 671-675 in File 12ShipsToken.sol

```
671     constructor (string memory name, string memory symbol, uint8 decimals) public {
672         _name = name;
673         _symbol = symbol;
674         _decimals = decimals;
675     }
```

The code meets the specification

Formal Verification Request 87

ERC20Detailed name correctness

13, May 2019
6.41 ms

Line 680-682 in File 12ShipsToken.sol

```
680     /*@CTK "ERC20Detailed name correctness"
681       @post __return == _name
682      */
```

Line 683-685 in File 12ShipsToken.sol

```
683     function name() public view returns (string memory) {
684         return _name;
685     }
```

The code meets the specification

Formal Verification Request 88

ERC20Detailed symbol correctness

13, May 2019
6.36 ms

Line 690-692 in File 12ShipsToken.sol

```
690     /*@CTK "ERC20Detailed symbol correctness"
691       @post __return == _symbol
692      */
```

Line 693-695 in File 12ShipsToken.sol

```
693     function symbol() public view returns (string memory) {
694         return _symbol;
695     }
```

The code meets the specification

Formal Verification Request 89

ERC20Detailed decimals correctness

13, May 2019
5.47 ms

Line 700-702 in File 12ShipsToken.sol

```
700     /*@CTK "ERC20Detailed decimals correctness"
701       @post __return == _decimals
702      */
```

Line 703-705 in File 12ShipsToken.sol

```
703     function decimals() public view returns (uint8) {
704         return _decimals;
705     }
```

The code meets the specification

Formal Verification Request 90

If method completes, integer overflow would not happen.

13, May 2019
143.54 ms

Line 789 in File 12ShipsToken.sol

```
789     //@CTK NO_OVERFLOW
```

Line 797-802 in File 12ShipsToken.sol

```
797     function freezeAccount(address holder) public onlyPauser returns (bool) {
798         require(!frozenAccount[holder]);
799         frozenAccount[holder] = true;
800         emit Freeze(holder);
801         return true;
802     }
```

The code meets the specification

Formal Verification Request 91

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

13, May 2019
5.07 ms

Line 790 in File 12ShipsToken.sol

```
790     //@CTK NO_BUF_OVERFLOW
```

Line 797-802 in File 12ShipsToken.sol

```
797     function freezeAccount(address holder) public onlyPauser returns (bool) {
798         require(!frozenAccount[holder]);
799         frozenAccount[holder] = true;
800         emit Freeze(holder);
801         return true;
802     }
```

The code meets the specification

Formal Verification Request 92

Method will not encounter an assertion failure.

13, May 2019
4.94 ms

Line 791 in File 12ShipsToken.sol

```
791     //@CTK NO_ASF
```

Line 797-802 in File 12ShipsToken.sol

```
797     function freezeAccount(address holder) public onlyPauser returns (bool) {
798         require(!frozenAccount[holder]);
799         frozenAccount[holder] = true;
800         emit Freeze(holder);
801         return true;
802     }
```

The code meets the specification

Formal Verification Request 93

freezeAccount correctness

13, May 2019
7.32 ms

Line 792-796 in File 12ShipsToken.sol

```
792     /*@CTK "freezeAccount correctness"
793       @post frozenAccount[holder] -> __reverted
794       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
795       @post !__reverted -> __post.frozenAccount[holder] == true
796      */
```

Line 797-802 in File 12ShipsToken.sol

```
797     function freezeAccount(address holder) public onlyPauser returns (bool) {
798         require(!frozenAccount[holder]);
799         frozenAccount[holder] = true;
800         emit Freeze(holder);
801         return true;
802     }
```

The code meets the specification

Formal Verification Request 94

If method completes, integer overflow would not happen.

13, May 2019
96.7 ms

Line 804 in File 12ShipsToken.sol

```
804     //@CTK NO_OVERFLOW
```

Line 812-817 in File 12ShipsToken.sol

```
812     function unfreezeAccount(address holder) public onlyPauser returns (bool) {
813         require(frozenAccount[holder]);
814         frozenAccount[holder] = false;
815         emit Unfreeze(holder);
816         return true;
817     }
```

The code meets the specification

Formal Verification Request 95

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

13, May 2019
5.25 ms

Line 805 in File 12ShipsToken.sol

```
805     //@CTK NO_BUF_OVERFLOW
```

Line 812-817 in File 12ShipsToken.sol

```
812     function unfreezeAccount(address holder) public onlyPauser returns (bool) {
813         require(frozenAccount[holder]);
814         frozenAccount[holder] = false;
815         emit Unfreeze(holder);
816         return true;
817     }
```

The code meets the specification

Formal Verification Request 96

Method will not encounter an assertion failure.

13, May 2019
4.99 ms

Line 806 in File 12ShipsToken.sol

```
806     //@CTK NO_ASF
```

Line 812-817 in File 12ShipsToken.sol

```
812     function unfreezeAccount(address holder) public onlyPauser returns (bool) {
813         require(frozenAccount[holder]);
814         frozenAccount[holder] = false;
815         emit Unfreeze(holder);
816         return true;
817     }
```

The code meets the specification

Formal Verification Request 97

unfreezeAccount correctness

13, May 2019
7.47 ms

Line 807-811 in File 12ShipsToken.sol

```
807     /*@CTK "unfreezeAccount correctness"
808       @post !frozenAccount[holder] -> __reverted
809       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
810       @post !__reverted -> __post.frozenAccount[holder] == false
811      */
```

Line 812-817 in File 12ShipsToken.sol

```
812     function unfreezeAccount(address holder) public onlyPauser returns (bool) {
813         require(frozenAccount[holder]);
814         frozenAccount[holder] = false;
815         emit Unfreeze(holder);
816         return true;
817     }
```

The code meets the specification

Formal Verification Request 98

If method completes, integer overflow would not happen.

13, May 2019
336.99 ms

Line 819 in File 12ShipsToken.sol

```
819     //@CTK FAIL NO_OVERFLOW
```

Line 830-836 in File 12ShipsToken.sol

```
830     function lock(address holder, uint256 value, uint256 releaseTime) public
            onlyPauser returns (bool) {
831         require(_balances[holder] >= value,"There is not enough balances of holder.");
832         _lock(holder,value,releaseTime);
833
834
835         return true;
836     }
```

This code violates the specification

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          holder = 0
5          releaseTime = 0
6          value = 0
7      }
8      This = 0
9      Internal = {
10         __has_assertion_failure = false
11         __has_buf_overflow = false
12         __has_overflow = false
13         __has_returned = false
14         __reverted = false
15         msg = {
16           "gas": 0,
17           "sender": 1,
18           "value": 0
19         }
20     }
21     Other = {
22         __return = false
23         block = {
24           "number": 0,
25           "timestamp": 0
26         }
27     }
28     Address_Map = [
29       {
30         "key": 0,
31         "value": {
32           "contract_name": "TWSToken",
```

```
        "balance": 0,
        "contract": {
          "implementation": 0,
          "timelockList": [
            {
              "key": 0,
              "value": [
                {
                  "key": 255,
                  "value": {
                    "_releaseTime": 8,
                    "_amount": 0
                  }
                },
                {
                  "key": "ALL_OTHERS",
                  "value": {
                    "_releaseTime": 0,
                    "_amount": 0
                  }
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": []
            }
          ],
          "frozenAccount": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_paused": false,
          "_pausers": {
            "bearer": [
              {
                "key": 1,
                "value": true
              },
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          },
          "owner": 0,
          "newOwner": 0,
          "_balances": [
            {
              "key": 0,
              "value": 224
            },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_name": "",
          "_symbol": "",
          "_decimals": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      holder = 0
      releaseTime = 0
      value = 0
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = true
      __has_returned = true
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 1,
        "value": 0
      }
  }
  Other = {
      __return = true
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
```

```
      "value": {
        "contract_name": "TWSToken",
        "balance": 0,
        "contract": {
          "implementation": 0,
          "timelockList": [
            {
              "key": 0,
              "value": []
            },
            {
              "key": "ALL_OTHERS",
              "value": []
            }
          ],
          "frozenAccount": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "_paused": false,
          "_pausers": {
            "bearer": [
              {
                "key": 1,
                "value": true
              },
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          },
          "owner": 0,
          "newOwner": 0,
          "_balances": [
            {
              "key": 0,
              "value": 224
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "_allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
```

```
              ]
            }
          ],
          "_totalSupply": 0,
          "_name": "",
          "_symbol": "",
          "_decimals": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 99

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

13, May 2019
8.92 ms

Line 820 in File 12ShipsToken.sol

```
820     //@CTK NO_BUF_OVERFLOW
```

Line 830-836 in File 12ShipsToken.sol

```
830     function lock(address holder, uint256 value, uint256 releaseTime) public
            onlyPauser returns (bool) {
831         require(_balances[holder] >= value,"There is not enough balances of holder.");
832         _lock(holder,value,releaseTime);
833
834
835         return true;
836     }
```

The code meets the specification

Formal Verification Request 100

Method will not encounter an assertion failure.

13, May 2019
8.42 ms

Line 821 in File 12ShipsToken.sol

```
821     //@CTK NO_ASF
```

Line 830-836 in File 12ShipsToken.sol

```
830     function lock(address holder, uint256 value, uint256 releaseTime) public
            onlyPauser returns (bool) {
831         require(_balances[holder] >= value,"There is not enough balances of holder.");
832         _lock(holder,value,releaseTime);
```

```
833
834
835         return true;
836     }
```

The code meets the specification

Formal Verification Request 101

lock correctness

13, May 2019
170.79 ms

Line 822-829 in File 12ShipsToken.sol

```
822     /*@CTK "lock correctness"
823       @post _balances[holder] < value -> __reverted
824       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
825       @post !__reverted -> __post._balances[holder] == _balances[holder] - value
826       @post !__reverted -> __post.timelockList[holder].length == timelockList[holder].
              length + 1
827       @post !__reverted -> __post.timelockList[holder][timelockList[holder].length].
              _amount == value
828       @post !__reverted -> __post.timelockList[holder][timelockList[holder].length].
              _releaseTime == releaseTime
829      */
```

Line 830-836 in File 12ShipsToken.sol

```
830     function lock(address holder, uint256 value, uint256 releaseTime) public
            onlyPauser returns (bool) {
831         require(_balances[holder] >= value,"There is not enough balances of holder.");
832         _lock(holder,value,releaseTime);
833
834
835         return true;
836     }
```

The code meets the specification

Formal Verification Request 102

If method completes, integer overflow would not happen.

13, May 2019
387.26 ms

Line 844 in File 12ShipsToken.sol

```
844     //@CTK NO_OVERFLOW
```

Line 854-858 in File 12ShipsToken.sol

```
854     function unlock(address holder, uint256 idx) public onlyPauser returns (bool) {
855         require( timelockList[holder].length > idx, "There is not lock info.");
856         _unlock(holder,idx);
```

```
857         return true;
858     }
```

The code meets the specification

Formal Verification Request 103

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

13, May 2019
154.85 ms

Line 845 in File 12ShipsToken.sol

```
845     //@CTK NO_BUF_OVERFLOW
```

Line 854-858 in File 12ShipsToken.sol

```
854     function unlock(address holder, uint256 idx) public onlyPauser returns (bool) {
855         require( timelockList[holder].length > idx, "There is not lock info.");
856         _unlock(holder,idx);
857         return true;
858     }
```

The code meets the specification

Formal Verification Request 104

Method will not encounter an assertion failure.

13, May 2019
14.32 ms

Line 846 in File 12ShipsToken.sol

```
846     //@CTK NO_ASF
```

Line 854-858 in File 12ShipsToken.sol

```
854     function unlock(address holder, uint256 idx) public onlyPauser returns (bool) {
855         require( timelockList[holder].length > idx, "There is not lock info.");
856         _unlock(holder,idx);
857         return true;
858     }
```

The code meets the specification

Formal Verification Request 105

unlock correctness

13, May 2019
651.36 ms

Line 847-853 in File 12ShipsToken.sol

```
847     /*@CTK "unlock correctness"
848       @post timelockList[holder].length <= idx -> __reverted
849       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
850       @post !__reverted -> __post._balances[holder] == _balances[holder] +
              timelockList[holder][idx]._amount
851       @post !__reverted && timelockList[holder].length > 0
852             -> __post.timelockList[holder].length == timelockList[holder].length - 1
853      */
```

Line 854-858 in File 12ShipsToken.sol

```
854     function unlock(address holder, uint256 idx) public onlyPauser returns (bool) {
855         require( timelockList[holder].length > idx, "There is not lock info.");
856         _unlock(holder,idx);
857         return true;
858     }
```

The code meets the specification

Formal Verification Request 106

If method completes, integer overflow would not happen.

13, May 2019
44.62 ms

Line 864 in File 12ShipsToken.sol

```
864     //@CTK NO_OVERFLOW
```

Line 873-876 in File 12ShipsToken.sol

```
873     function upgradeTo(address _newImplementation) public onlyOwner {
874         require(implementation != _newImplementation);
875         _setImplementation(_newImplementation);
876     }
```

The code meets the specification

Formal Verification Request 107

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

13, May 2019
3.4 ms

Line 865 in File 12ShipsToken.sol

```
865     //@CTK NO_BUF_OVERFLOW
```

Line 873-876 in File 12ShipsToken.sol

```
873     function upgradeTo(address _newImplementation) public onlyOwner {
874         require(implementation != _newImplementation);
875         _setImplementation(_newImplementation);
876     }
```

The code meets the specification

Formal Verification Request 108

Method will not encounter an assertion failure.

13, May 2019
3.08 ms

Line 866 in File 12ShipsToken.sol

```
866     //@CTK NO_ASF
```

Line 873-876 in File 12ShipsToken.sol

```
873     function upgradeTo(address _newImplementation) public onlyOwner {
874         require(implementation != _newImplementation);
875         _setImplementation(_newImplementation);
876     }
```

The code meets the specification

Formal Verification Request 109

upgradeTo correctness

13, May 2019
9.11 ms

Line 867-872 in File 12ShipsToken.sol

```
867     /*@CTK "upgradeTo correctness"
868       @post implementation == _newImplementation -> __reverted
869       @post msg.sender != owner -> __reverted
870       @post implementation != _newImplementation && msg.sender == owner
871             -> __post.implementation == _newImplementation
872      */
```

Line 873-876 in File 12ShipsToken.sol

```
873     function upgradeTo(address _newImplementation) public onlyOwner {
874         require(implementation != _newImplementation);
875         _setImplementation(_newImplementation);
876     }
```

The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File 12ShipsToken.sol

```
1 pragma solidity ^0.5.0;
```

Only these compiler versions are safe to compile your code: 0.5.0, 0.5.1, 0.5.2, 0.5.3, 0.5.4, 0.5.6

TIMESTAMP_DEPENDENCY

Line 907 in File 12ShipsToken.sol

```
907             if (timelockList[holder][idx]._releaseTime <= now) {
```

"now" can be influenced by minors to some degree

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **12ShipsToken.sol** `4d744286683ce994d29fa0024881e557f137647c9a99dc1bff7587ca1e5c4fa1`

Summary

CertiK team is invited by The 12Ships 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 checking as well as manual reviews by smart contract experts. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and 12Ships team has been actively giving us updates for the source code and feedback about the business logics.

At this point the 12Ships team didn't provide other repositories sources as testing and documentation reference. We recommend having more unit tests coverage together with documentation to simulate potential use cases and walk through the functionalities to token holders, especially those super admin privileges that may impact the decentralized nature. Though the token itself fulfilled the interfaces of ERC20, many extra functionalities were added (i.e. `timelockList related actions`), which introduced much more complexity and decreased readability. We recommend to decouple such features into different components so as to have a much cleaner token contract without any add-ons.

Overall we found the `12ShipsToken.sol` contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrative 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 anti-patterns 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.

12ShipsToken.sol

- `div(uint256 a, uint256 b)` – `uint` is considered non-negative, prefer `require(b != 0);`.
- `_unlock(address holder, uint256 idx)` – prefer to use `memory` instead of `storage`, or simply `timelockList[holder][idx].amount`.
- `lock(address holder, uint256 value, uint256 releaseTime)` – ensure the releaseTime that is always later than current time, i.e `require (releaseTime > now, 'release time should be a future time')`.

Source Code with CertiK Labels

File 12ShipsToken.sol

```
1  pragma solidity ^0.5.0;
2
3  library SafeMath {
4      /**
5      * @dev Multiplies two unsigned integers, reverts on overflow.
6      */
7      /*@CTK "SafeMath mul"
8        @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
9        @post !__reverted -> __return == a * b
10       @post !__reverted == !__has_overflow
11       @post !(__has_buf_overflow)
12       @post !(__has_assertion_failure)
13      */
14     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
15         // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
16         // benefit is lost if 'b' is also tested.
17         // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
18         if (a == 0) {
19             return 0;
20         }
21
22         uint256 c = a * b;
23         require(c / a == b);
24
25         return c;
26     }
27
28     /**
29     * @dev Integer division of two unsigned integers truncating the quotient, reverts
           on division by zero.
30     */
31     /*@CTK "SafeMath div"
32       @post b != 0 -> !__reverted
33       @post !__reverted -> __return == a / b
34       @post !__reverted -> !__has_overflow
35       @post !(__has_buf_overflow)
36       @post !(__has_assertion_failure)
37     */
38     function div(uint256 a, uint256 b) internal pure returns (uint256) {
39         // Solidity only automatically asserts when dividing by 0
40         require(b > 0);
41         uint256 c = a / b;
42         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
43
44         return c;
45     }
46
47     /**
48     * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is
           greater than minuend).
49     */
50     /*@CTK "SafeMath sub"
51       @post (a < b) == __reverted
52       @post !__reverted -> __return == a - b
```

```
53        @post !__reverted -> !__has_overflow
54        @post !(__has_buf_overflow)
55        @post !(__has_assertion_failure)
56      */
57      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
58          require(b <= a);
59          uint256 c = a - b;
60
61          return c;
62      }
63
64      /**
65      * @dev Adds two unsigned integers, reverts on overflow.
66      */
67      /*@CTK "SafeMath add"
68        @post (a + b < a || a + b < b) == __reverted
69        @post !__reverted -> __return == a + b
70        @post !__reverted -> !__has_overflow
71        @post !(__has_buf_overflow)
72        @post !(__has_assertion_failure)
73      */
74      function add(uint256 a, uint256 b) internal pure returns (uint256) {
75          uint256 c = a + b;
76          require(c >= a);
77
78          return c;
79      }
80
81      /**
82      * @dev Divides two unsigned integers and returns the remainder (unsigned integer
            modulo),
83      * reverts when dividing by zero.
84      */
85      /*@CTK "SafeMath div"
86        @post b != 0 -> !__reverted
87        @post !__reverted -> __return == a % b
88        @post !__reverted -> !__has_overflow
89        @post !(__has_buf_overflow)
90        @post !(__has_assertion_failure)
91       */
92      function mod(uint256 a, uint256 b) internal pure returns (uint256) {
93          require(b != 0);
94          return a % b;
95      }
96  }
97
98  library Roles {
99      struct Role {
100         mapping (address => bool) bearer;
101     }
102
103     /**
104      * @dev give an account access to this role
105      */
106     //@CTK NO_OVERFLOW
107     //@CTK NO_BUF_OVERFLOW
108     //@CTK NO_ASF
109     /*@CTK "Roles add correctness"
```

```
110       @post account == 0x0 -> __reverted
111       @post role.bearer[account] -> __reverted
112       @post account != 0x0 && !role.bearer[account] -> !__reverted
113      */
114     function add(Role storage role, address account) internal {
115         require(account != address(0));
116         require(!role.bearer[account]);
117 
118         role.bearer[account] = true;
119     }
120 
121     /**
122      * @dev remove an account's access to this role
123      */
124     //@CTK NO_OVERFLOW
125     //@CTK NO_BUF_OVERFLOW
126     //@CTK NO_ASF
127     /*@CTK "Roles add correctness"
128       @post account == 0x0 -> __reverted
129       @post !role.bearer[account] -> __reverted
130       @post account != 0x0 && role.bearer[account] -> !__reverted
131      */
132     function remove(Role storage role, address account) internal {
133         require(account != address(0));
134         require(role.bearer[account]);
135 
136         role.bearer[account] = false;
137     }
138 
139     /**
140      * @dev check if an account has this role
141      * @return bool
142      */
143     //@CTK NO_OVERFLOW
144     //@CTK NO_BUF_OVERFLOW
145     //@CTK NO_ASF
146     /*@CTK "Roles has correctness"
147       @post account == 0x0 -> __reverted
148       @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
149      */
150     function has(Role storage role, address account) internal view returns (bool) {
151         require(account != address(0));
152         return role.bearer[account];
153     }
154 }
155 
156 contract Ownable {
157     address public owner;
158     address public newOwner;
159 
160     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
        );
161 
162     constructor() public {
163         owner = msg.sender;
164         newOwner = address(0);
165     }
166 
```

```
167     modifier onlyOwner() {
168         require(msg.sender == owner);
169         _;
170     }
171     modifier onlyNewOwner() {
172         require(msg.sender != address(0));
173         require(msg.sender == newOwner);
174         _;
175     }
176
177     function isOwner(address account) public view returns (bool) {
178         if( account == owner ){
179             return true;
180         }
181         else {
182             return false;
183         }
184     }
185
186     function transferOwnership(address _newOwner) public onlyOwner {
187         require(_newOwner != address(0));
188         newOwner = _newOwner;
189     }
190
191     function acceptOwnership() public onlyNewOwner returns(bool) {
192         emit OwnershipTransferred(owner, newOwner);
193         owner = newOwner;
194         newOwner = address(0);
195     }
196 }
197
198 contract PauserRole is Ownable{
199     using Roles for Roles.Role;
200
201     event PauserAdded(address indexed account);
202     event PauserRemoved(address indexed account);
203
204     Roles.Role private _pausers;
205
206     //@CTK NO_OVERFLOW
207     //@CTK NO_BUF_OVERFLOW
208     //@CTK NO_ASF
209     /*@CTK "PauserRole constructor correctness"
210       @post msg.sender == 0x0 -> __reverted
211       @post _pausers.bearer[msg.sender] -> __reverted
212       @post msg.sender != 0x0 && !_pausers.bearer[msg.sender]
213           -> !__reverted && __post._pausers.bearer[msg.sender]
214      */
215     constructor () internal {
216         _addPauser(msg.sender);
217     }
218
219     modifier onlyPauser() {
220         require(isPauser(msg.sender)|| isOwner(msg.sender));
221         _;
222     }
223
224     //@CTK NO_OVERFLOW
```

```
225     //@CTK NO_BUF_OVERFLOW
226     //@CTK NO_ASF
227     /*@CTK "isBurner correctness"
228       @post account == 0x0 -> __reverted
229       @post account != 0x0 -> !__reverted && __return == _pausers.bearer[account]
230      */
231     function isPauser(address account) public view returns (bool) {
232         return _pausers.has(account);
233     }
234
235     //@CTK NO_OVERFLOW
236     //@CTK NO_BUF_OVERFLOW
237     //@CTK NO_ASF
238     /*@CTK "addPauser correctness"
239       @post account == 0x0 -> __reverted
240       @post msg.sender == 0x0 -> __reverted
241       @post _pausers.bearer[account] -> __reverted
242       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
243       @post account != 0x0 && !_pausers.bearer[account]
244           && msg.sender != 0x0 && _pausers.bearer[msg.sender]
245           -> !__reverted && __post._pausers.bearer[account]
246       @post account != 0x0 && !_pausers.bearer[account]
247           && msg.sender != 0x0 && owner == msg.sender
248           -> !__reverted && __post._pausers.bearer[account]
249      */
250     function addPauser(address account) public onlyPauser {
251         _addPauser(account);
252     }
253
254     //@CTK NO_OVERFLOW
255     //@CTK NO_BUF_OVERFLOW
256     //@CTK NO_ASF
257     /*@CTK "removePauser correctness"
258       @post account == 0x0 -> __reverted
259       @post !_pausers.bearer[account] -> __reverted
260       @post owner != msg.sender -> __reverted
261       @post account != 0x0 && _pausers.bearer[account] && owner == msg.sender
262           -> !__reverted && !__post._pausers.bearer[account]
263      */
264     function removePauser(address account) public onlyOwner {
265         _removePauser(account);
266     }
267
268     //@CTK NO_OVERFLOW
269     //@CTK NO_BUF_OVERFLOW
270     //@CTK NO_ASF
271     /*@CTK "renouncePauser correctness"
272       @post msg.sender == 0x0 -> __reverted
273       @post !_pausers.bearer[msg.sender] -> __reverted
274       @post msg.sender != 0x0 && _pausers.bearer[msg.sender]
275           -> !__reverted && !__post._pausers.bearer[msg.sender]
276      */
277     function renouncePauser() public {
278         _removePauser(msg.sender);
279     }
280
281     function _addPauser(address account) internal {
282         _pausers.add(account);
```

```
283         emit PauserAdded(account);
284     }
285
286     function _removePauser(address account) internal {
287         _pausers.remove(account);
288         emit PauserRemoved(account);
289     }
290 }
291
292 contract Pausable is PauserRole {
293     event Paused(address account);
294     event Unpaused(address account);
295
296     bool private _paused;
297
298     //@CTK NO_OVERFLOW
299     //@CTK NO_BUF_OVERFLOW
300     //@CTK NO_ASF
301     /*@CTK "Pausable constructor correctness"
302       @post __post._paused == false
303      */
304     constructor () internal {
305         _paused = false;
306     }
307
308     /**
309      * @return true if the contract is paused, false otherwise.
310      */
311     //@CTK NO_OVERFLOW
312     //@CTK NO_BUF_OVERFLOW
313     //@CTK NO_ASF
314     /*@CTK "Pausable paused correctness"
315       @post __return == _paused
316      */
317     function paused() public view returns (bool) {
318         return _paused;
319     }
320
321     /**
322      * @dev Modifier to make a function callable only when the contract is not paused.
323      */
324     modifier whenNotPaused() {
325         require(!_paused);
326         _;
327     }
328
329     /**
330      * @dev Modifier to make a function callable only when the contract is paused.
331      */
332     modifier whenPaused() {
333         require(_paused);
334         _;
335     }
336
337     /**
338      * @dev called by the owner to pause, triggers stopped state
339      */
340     //@CTK NO_OVERFLOW
```

```
341     //@CTK NO_BUF_OVERFLOW
342     //@CTK NO_ASF
343     /*@CTK "Pausable pause correctness"
344       @post _paused -> __reverted
345       @post msg.sender == 0x0 -> __reverted
346       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
347       @post !_paused && msg.sender != 0x0 && _pausers.bearer[msg.sender]
348           -> __post._paused
349       @post !_paused && msg.sender != 0x0 && owner == msg.sender
350           -> __post._paused
351     */
352     function pause() public onlyPauser whenNotPaused {
353         _paused = true;
354         emit Paused(msg.sender);
355     }
356
357     /**
358      * @dev called by the owner to unpause, returns to normal state
359      */
360     //@CTK NO_OVERFLOW
361     //@CTK NO_BUF_OVERFLOW
362     //@CTK NO_ASF
363     /*@CTK "Pausable unpause correctness"
364       @post !_paused -> __reverted
365       @post msg.sender == 0x0 -> __reverted
366       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
367       @post _paused && msg.sender != 0x0 && _pausers.bearer[msg.sender]
368           -> !__post._paused
369       @post _paused && msg.sender != 0x0 && owner == msg.sender
370           -> !__post._paused
371     */
372     function unpause() public onlyPauser whenPaused {
373         _paused = false;
374         emit Unpaused(msg.sender);
375     }
376 }
377
378 interface IERC20 {
379     function transfer(address to, uint256 value) external returns (bool);
380
381     function approve(address spender, uint256 value) external returns (bool);
382
383     function transferFrom(address from, address to, uint256 value) external returns (
        bool);
384
385     function totalSupply() external view returns (uint256);
386
387     function balanceOf(address who) external view returns (uint256);
388
389     function allowance(address owner, address spender) external view returns (uint256)
        ;
390
391     event Transfer(address indexed from, address indexed to, uint256 value);
392
393     event Approval(address indexed owner, address indexed spender, uint256 value);
394 }
395
396 contract ERC20 is IERC20 {
```

```
397     using SafeMath for uint256;
398
399     mapping (address => uint256) internal _balances;
400
401     mapping (address => mapping (address => uint256)) internal _allowed;
402
403     uint256 private _totalSupply;
404
405     /**
406     * @dev Total number of tokens in existence
407     */
408     //@CTK NO_OVERFLOW
409     //@CTK NO_BUF_OVERFLOW
410     //@CTK NO_ASF
411     /*@CTK "totalSupply correctness"
412       @post __return == _totalSupply
413      */
414     function totalSupply() public view returns (uint256) {
415         return _totalSupply;
416     }
417
418     /**
419     * @dev Gets the balance of the specified address.
420     * @param owner The address to query the balance of.
421     * @return An uint256 representing the amount owned by the passed address.
422     */
423     //@CTK NO_OVERFLOW
424     //@CTK NO_BUF_OVERFLOW
425     //@CTK NO_ASF
426     /*@CTK "balanceOf correctness"
427       @post __return == _balances[owner]
428      */
429     function balanceOf(address owner) public view returns (uint256) {
430         return _balances[owner];
431     }
432
433     /**
434      * @dev Function to check the amount of tokens that an owner allowed to a spender.
435      * @param owner address The address which owns the funds.
436      * @param spender address The address which will spend the funds.
437      * @return A uint256 specifying the amount of tokens still available for the
           spender.
438      */
439     //@CTK NO_OVERFLOW
440     //@CTK NO_BUF_OVERFLOW
441     //@CTK NO_ASF
442     /*@CTK "allowance correctness"
443       @post __return == _allowed[owner][spender]
444      */
445     function allowance(address owner, address spender) public view returns (uint256) {
446         return _allowed[owner][spender];
447     }
448
449     /**
450     * @dev Transfer token for a specified address
451     * @param to The address to transfer to.
452     * @param value The amount to be transferred.
453     */
```

```
454     //@CTK NO_OVERFLOW
455     //@CTK NO_BUF_OVERFLOW
456     //@CTK NO_ASF
457     /*@CTK "transfer correctness"
458       @tag assume_completion
459       @post to != 0x0
460       @post value <= _balances[msg.sender]
461       @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - value
462       @post to != msg.sender -> __post._balances[to] == _balances[to] + value
463       @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
464      */
465     function transfer(address to, uint256 value) public returns (bool) {
466         _transfer(msg.sender, to, value);
467         return true;
468     }
469
470     /**
471      * @dev Approve the passed address to spend the specified amount of tokens on
             behalf of msg.sender.
472      * Beware that changing an allowance with this method brings the risk that someone
              may use both the old
473      * and the new allowance by unfortunate transaction ordering. One possible
             solution to mitigate this
474      * race condition is to first reduce the spender's allowance to 0 and set the
             desired value afterwards:
475      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
476      * @param spender The address which will spend the funds.
477      * @param value The amount of tokens to be spent.
478      */
479     //@CTK NO_OVERFLOW
480     //@CTK NO_BUF_OVERFLOW
481     //@CTK NO_ASF
482     /*@CTK "approve correctness"
483       @post spender == 0x0 -> __reverted
484       @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
485      */
486     function approve(address spender, uint256 value) public returns (bool) {
487         require(spender != address(0));
488
489         _allowed[msg.sender][spender] = value;
490         emit Approval(msg.sender, spender, value);
491         return true;
492     }
493
494     /**
495      * @dev Transfer tokens from one address to another.
496      * Note that while this function emits an Approval event, this is not required as
             per the specification,
497      * and other compliant implementations may not emit the event.
498      * @param from address The address which you want to send tokens from
499      * @param to address The address which you want to transfer to
500      * @param value uint256 the amount of tokens to be transferred
501      */
502     //@CTK NO_OVERFLOW
503     //@CTK NO_BUF_OVERFLOW
504     //@CTK NO_ASF
505     /*@CTK "transferFrom correctness"
```

```
506       @tag assume_completion
507       @post to != 0x0
508       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
509       @post to != from -> __post._balances[from] == _balances[from] - value
510       @post to != from -> __post._balances[to] == _balances[to] + value
511       @post to == from -> __post._balances[from] == _balances[from]
512       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
513     */
514     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
515         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
516         _transfer(from, to, value);
517         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
518         return true;
519     }
520
521     /**
522      * @dev Increase the amount of tokens that an owner allowed to a spender.
523      * approve should be called when allowed_[_spender] == 0. To increment
524      * allowed value is better to use this function to avoid 2 calls (and wait until
525      * the first transaction is mined)
526      * From MonolithDAO Token.sol
527      * Emits an Approval event.
528      * @param spender The address which will spend the funds.
529      * @param addedValue The amount of tokens to increase the allowance by.
530      */
531     //@CTK NO_OVERFLOW
532     //@CTK NO_BUF_OVERFLOW
533     //@CTK NO_ASF
534     /*@CTK "increaseAllowance correctness"
535       @tag assume_completion
536       @post spender != 0x0
537       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
              addedValue
538     */
539     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
540         require(spender != address(0));
541
542         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
543         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
544         return true;
545     }
546
547     /**
548      * @dev Decrease the amount of tokens that an owner allowed to a spender.
549      * approve should be called when allowed_[_spender] == 0. To decrement
550      * allowed value is better to use this function to avoid 2 calls (and wait until
551      * the first transaction is mined)
552      * From MonolithDAO Token.sol
553      * Emits an Approval event.
554      * @param spender The address which will spend the funds.
555      * @param subtractedValue The amount of tokens to decrease the allowance by.
556      */
557     //@CTK NO_OVERFLOW
558     //@CTK NO_BUF_OVERFLOW
559     //@CTK NO_ASF
560     /*@CTK "decreaseAllowance correctness"
```

```
561       @tag assume_completion
562       @post spender != 0x0
563       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
564      */
565     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
566         require(spender != address(0));
567
568         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
569         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
570         return true;
571     }
572
573     /**
574     * @dev Transfer token for a specified addresses
575     * @param from The address to transfer from.
576     * @param to The address to transfer to.
577     * @param value The amount to be transferred.
578     */
579     function _transfer(address from, address to, uint256 value) internal {
580         require(to != address(0));
581
582         _balances[from] = _balances[from].sub(value);
583         _balances[to] = _balances[to].add(value);
584         emit Transfer(from, to, value);
585     }
586
587     /**
588      * @dev Internal function that mints an amount of the token and assigns it to
589      * an account. This encapsulates the modification of balances such that the
590      * proper events are emitted.
591      * @param account The account that will receive the created tokens.
592      * @param value The amount that will be created.
593      */
594     function _mint(address account, uint256 value) internal {
595         require(account != address(0));
596
597         _totalSupply = _totalSupply.add(value);
598         _balances[account] = _balances[account].add(value);
599         emit Transfer(address(0), account, value);
600     }
601
602     /**
603      * @dev Internal function that burns an amount of the token of a given
604      * account.
605      * @param account The account whose tokens will be burnt.
606      * @param value The amount that will be burnt.
607      */
608     function _burn(address account, uint256 value) internal {
609         require(account != address(0));
610
611         _totalSupply = _totalSupply.sub(value);
612         _balances[account] = _balances[account].sub(value);
613         emit Transfer(account, address(0), value);
614     }
615
```

```
616     /**
617      * @dev Internal function that burns an amount of the token of a given
618      * account, deducting from the sender's allowance for said account. Uses the
619      * internal burn function.
620      * Emits an Approval event (reflecting the reduced allowance).
621      * @param account The account whose tokens will be burnt.
622      * @param value The amount that will be burnt.
623      */
624     function _burnFrom(address account, uint256 value) internal {
625         _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
626         _burn(account, value);
627         emit Approval(account, msg.sender, _allowed[account][msg.sender]);
628     }
629 }
630
631
632
633 contract ERC20Pausable is ERC20, Pausable {
634     function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
635         return super.transfer(to, value);
636     }
637
638     function transferFrom(address from, address to, uint256 value) public
            whenNotPaused returns (bool) {
639         return super.transferFrom(from, to, value);
640     }
641
642     /*
643      * approve/increaseApprove/decreaseApprove can be set when Paused state
644      */
645
646     /*
647      * function approve(address spender, uint256 value) public whenNotPaused returns (
            bool) {
648      *     return super.approve(spender, value);
649      * }
650      *
651      * function increaseAllowance(address spender, uint addedValue) public
            whenNotPaused returns (bool success) {
652      *     return super.increaseAllowance(spender, addedValue);
653      * }
654      *
655      * function decreaseAllowance(address spender, uint subtractedValue) public
            whenNotPaused returns (bool success) {
656      *     return super.decreaseAllowance(spender, subtractedValue);
657      * }
658      */
659 }
660
661 contract ERC20Detailed is IERC20 {
662     string private _name;
663     string private _symbol;
664     uint8 private _decimals;
665
666     /*@CTK "ERC20Detailed constructor correctness"
667       @post __post._name == name
668       @post __post._symbol == symbol
669       @post __post._decimals == decimals
```

```
670      */
671     constructor (string memory name, string memory symbol, uint8 decimals) public {
672         _name = name;
673         _symbol = symbol;
674         _decimals = decimals;
675     }
676 
677     /**
678      * @return the name of the token.
679      */
680     /*@CTK "ERC20Detailed name correctness"
681       @post __return == _name
682      */
683     function name() public view returns (string memory) {
684         return _name;
685     }
686 
687     /**
688      * @return the symbol of the token.
689      */
690     /*@CTK "ERC20Detailed symbol correctness"
691       @post __return == _symbol
692      */
693     function symbol() public view returns (string memory) {
694         return _symbol;
695     }
696 
697     /**
698      * @return the number of decimals of the token.
699      */
700     /*@CTK "ERC20Detailed decimals correctness"
701       @post __return == _decimals
702      */
703     function decimals() public view returns (uint8) {
704         return _decimals;
705     }
706 }
707 
708 contract TWSToken is ERC20Detailed, ERC20Pausable {
709 
710     struct LockInfo {
711         uint256 _releaseTime;
712         uint256 _amount;
713     }
714 
715     address public implementation;
716 
717     mapping (address => LockInfo[]) public timelockList;
718   mapping (address => bool) public frozenAccount;
719 
720     event Freeze(address indexed holder);
721     event Unfreeze(address indexed holder);
722     event Lock(address indexed holder, uint256 value, uint256 releaseTime);
723     event Unlock(address indexed holder, uint256 value);
724 
725     modifier notFrozen(address _holder) {
726         require(!frozenAccount[_holder]);
727         _;
```

```
728     }
729
730     constructor() ERC20Detailed("12SHIPS TOKEN", "12SHP", 18) public {
731
732         _mint(msg.sender, 1000000000 * (10 ** 18));
733     }
734
735     /*CTK balanceOf
736       @tag assume_completion
737       @post __return >= _balances[owner]
738      */
739     function balanceOf(address owner) public view returns (uint256) {
740
741         uint256 totalBalance = super.balanceOf(owner);
742         if( timelockList[owner].length >0 ){
743             /*CTK Forloop_balanceOf
744               @inv i < timelockList[owner].length
745               @inv totalBalance >= totalBalance__pre + timelockList[owner][i]._amount
746               @post i == timelockList[owner].length
747               @post !__should_return
748              */
749             for(uint i=0; i<timelockList[owner].length;i++){
750                 totalBalance = totalBalance.add(timelockList[owner][i]._amount);
751             }
752         }
753
754         return totalBalance;
755     }
756
757
758     /*CTK transfer
759       @tag assume_completion
760       @post to != 0x0
761       @post value <= _balances[msg.sender]
762       @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - value
763       @post to != msg.sender -> __post._balances[to] == _balances[to] + value
764       @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
765      */
766     function transfer(address to, uint256 value) public notFrozen(msg.sender) returns
          (bool) {
767         if (timelockList[msg.sender].length > 0 ) {
768             _autoUnlock(msg.sender);
769         }
770         return super.transfer(to, value);
771     }
772
773     /*CTK transferFrom
774       @tag assume_completion
775       @post to != 0x0
776       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
777       @post to != from -> __post._balances[from] == _balances[from] - value
778       @post to != from -> __post._balances[to] == _balances[to] + value
779       @post to == from -> __post._balances[from] == _balances[from]
780       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
781      */
782     function transferFrom(address from, address to, uint256 value) public notFrozen(
          from) returns (bool) {
```

```
783         if (timelockList[from].length > 0) {
784             _autoUnlock(from);
785         }
786         return super.transferFrom(from, to, value);
787     }
788 
789     //@CTK NO_OVERFLOW
790     //@CTK NO_BUF_OVERFLOW
791     //@CTK NO_ASF
792     /*@CTK "freezeAccount correctness"
793       @post frozenAccount[holder] -> __reverted
794       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
795       @post !__reverted -> __post.frozenAccount[holder] == true
796      */
797     function freezeAccount(address holder) public onlyPauser returns (bool) {
798         require(!frozenAccount[holder]);
799         frozenAccount[holder] = true;
800         emit Freeze(holder);
801         return true;
802     }
803 
804     //@CTK NO_OVERFLOW
805     //@CTK NO_BUF_OVERFLOW
806     //@CTK NO_ASF
807     /*@CTK "unfreezeAccount correctness"
808       @post !frozenAccount[holder] -> __reverted
809       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
810       @post !__reverted -> __post.frozenAccount[holder] == false
811      */
812     function unfreezeAccount(address holder) public onlyPauser returns (bool) {
813         require(frozenAccount[holder]);
814         frozenAccount[holder] = false;
815         emit Unfreeze(holder);
816         return true;
817     }
818 
819     //@CTK FAIL NO_OVERFLOW
820     //@CTK NO_BUF_OVERFLOW
821     //@CTK NO_ASF
822     /*@CTK "lock correctness"
823       @post _balances[holder] < value -> __reverted
824       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
825       @post !__reverted -> __post._balances[holder] == _balances[holder] - value
826       @post !__reverted -> __post.timelockList[holder].length == timelockList[holder].
              length + 1
827       @post !__reverted -> __post.timelockList[holder][timelockList[holder].length].
              _amount == value
828       @post !__reverted -> __post.timelockList[holder][timelockList[holder].length].
              _releaseTime == releaseTime
829      */
830     function lock(address holder, uint256 value, uint256 releaseTime) public
          onlyPauser returns (bool) {
831         require(_balances[holder] >= value,"There is not enough balances of holder.");
832         _lock(holder,value,releaseTime);
833 
834 
835         return true;
836     }
```

```
837
838     function transferWithLock(address holder, uint256 value, uint256 releaseTime)
            public onlyPauser returns (bool) {
839         _transfer(msg.sender, holder, value);
840         _lock(holder,value,releaseTime);
841         return true;
842     }
843
844     //@CTK NO_OVERFLOW
845     //@CTK NO_BUF_OVERFLOW
846     //@CTK NO_ASF
847     /*@CTK "unlock correctness"
848       @post timelockList[holder].length <= idx -> __reverted
849       @post !_pausers.bearer[msg.sender] && owner != msg.sender -> __reverted
850       @post !__reverted -> __post._balances[holder] == _balances[holder] +
              timelockList[holder][idx]._amount
851       @post !__reverted && timelockList[holder].length > 0
852             -> __post.timelockList[holder].length == timelockList[holder].length - 1
853      */
854     function unlock(address holder, uint256 idx) public onlyPauser returns (bool) {
855         require( timelockList[holder].length > idx, "There is not lock info.");
856         _unlock(holder,idx);
857         return true;
858     }
859
860     /**
861      * @dev Upgrades the implementation address
862      * @param _newImplementation address of the new implementation
863      */
864     //@CTK NO_OVERFLOW
865     //@CTK NO_BUF_OVERFLOW
866     //@CTK NO_ASF
867     /*@CTK "upgradeTo correctness"
868       @post implementation == _newImplementation -> __reverted
869       @post msg.sender != owner -> __reverted
870       @post implementation != _newImplementation && msg.sender == owner
871             -> __post.implementation == _newImplementation
872      */
873     function upgradeTo(address _newImplementation) public onlyOwner {
874         require(implementation != _newImplementation);
875         _setImplementation(_newImplementation);
876     }
877
878     function _lock(address holder, uint256 value, uint256 releaseTime) internal
            returns(bool) {
879         _balances[holder] = _balances[holder].sub(value);
880         timelockList[holder].push( LockInfo(releaseTime, value) );
881
882         emit Lock(holder, value, releaseTime);
883         return true;
884     }
885
886     function _unlock(address holder, uint256 idx) internal returns(bool) {
887         LockInfo storage lockinfo = timelockList[holder][idx];
888         uint256 releaseAmount = lockinfo._amount;
889
890         delete timelockList[holder][idx];
```

```
891         timelockList[holder][idx] = timelockList[holder][timelockList[holder].length.
                sub(1)];
892         timelockList[holder].length -=1;
893 
894         emit Unlock(holder, releaseAmount);
895         _balances[holder] = _balances[holder].add(releaseAmount);
896 
897         return true;
898     }
899 
900     function _autoUnlock(address holder) internal returns(bool) {
901         /*CTK _autoUnlock
902           @inv idx < timelockList[holder].length
903           @post idx == timelockList[holder].length
904           @post !__should_return
905          */
906         for(uint256 idx =0; idx < timelockList[holder].length ; idx++ ) {
907             if (timelockList[holder][idx]._releaseTime <= now) {
908                 // If lockupinfo was deleted, loop restart at same position.
909                 if( _unlock(holder, idx) ) {
910                     idx -=1;
911                 }
912             }
913         }
914         return true;
915     }
916 
917     /**
918      * @dev Sets the address of the current implementation
919      * @param _newImp address of the new implementation
920      */
921     function _setImplementation(address _newImp) internal {
922         implementation = _newImp;
923     }
924 
925     /**
926      * @dev Fallback function allowing to perform a delegatecall
927      * to the given implementation. This function will return
928      * whatever the implementation call returns
929      */
930     function () payable external {
931         address impl = implementation;
932         require(impl != address(0));
933         assembly {
934             let ptr := mload(0x40)
935             calldatacopy(ptr, 0, calldatasize)
936             let result := delegatecall(gas, impl, ptr, calldatasize, 0, 0)
937             let size := returndatasize
938             returndatacopy(ptr, 0, size)
939 
940             switch result
941             case 0 { revert(ptr, size) }
942             default { return(ptr, size) }
943         }
944     }
945 }
```