

CERTIK AUDIT REPORT FOR hCAPTCHA



Request Date: 2019-05-24
Revision Date: 2019-07-03
Platform Name: Ethereum



Contents

| | |
|---------------------------------------|-----------|
| Disclaimer | 1 |
| About CertiK | 2 |
| Exective Summary | 3 |
| Vulnerability Classification | 3 |
| Testing Summary | 4 |
| Audit Score | 4 |
| Type of Issues | 4 |
| Vulnerability Details | 5 |
| Manual Review Notes | 6 |
| Static Analysis Results | 7 |
| Formal Verification Results | 8 |
| How to read | 8 |
| Source Code with CertiK Labels | 22 |

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 hCaptcha(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

About CertiK

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

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

For more information: <https://certik.org/>

Executive Summary

This report has been prepared as product of the Smart Contract Audit request by hCaptcha. This audit was conducted to discover issues and vulnerabilities in the source code of hCaptcha'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.

Jul 03, 2019



Type of Issues

CertiK smart label engine applied 100% covered 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

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **HMTOKEN.sol**
2beb2103decfe1bb07195b0546082ad5f12a38c3f3dec38382eb617ef11c150b
- **HMTOKENInterface.sol**
f93b10306b6d53835f10533dc941b4c7e49d2aa4e6149d154fcd36158d3abc6b
- **Migrations.sol**
b2b5280e1b16afb218c37a3083c6e8d2a2c82627ab27f9fe044af6f24e2008a8
- **SafeMath.sol**
95e6ccf2891bd7814fedb6768c99cc61d813814029f16e907d8a8c3c9c5c6c10

Summary

CertiK team is invited by the hCaptcha 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 hCaptcha team has been actively giving us updates for the source code and feedback about the business logic.

Overall we found the hCaptcha contracts follow good practices, with a reasonable amount of features on top of the ERC20 such as bulk functions to handle a batch of transfer or approve requests. 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.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity 0.5.9;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File HMToken.sol

```
1 pragma solidity 0.5.9;
```

 Only these compiler versions are safe to compile your code: 0.5.9

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 | <pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from][msg.sender] == 34 */ </pre> |
| Raw code location | Line 35-41 in File howtoread.sol |
| Raw code | <pre> 35 function transferFrom(address from, address to 36) { 37 balances[from] = balances[from].sub(tokens 38 allowed[from][msg.sender] = allowed[from][39 balances[to] = balances[to].add(tokens); 40 emit Transfer(from, to, tokens); 41 return true; </pre> |
| Counterexample | <div>  This code violates the specification </div> |
| Initial environment | <pre> 1 Counter Example: 2 Before Execution: 3 Input = { 4 from = 0x0 5 to = 0x0 6 tokens = 0x6c 7 } 8 This = 0 </pre> |
| Post environment | <pre> 52 } 53 balance: 0x0 54 } 55 } 56 57 After Execution: 58 Input = { 59 from = 0x0 60 to = 0x0 61 tokens = 0x6c </pre> |

Formal Verification Request 1

SafeMath add

 03, Jul 2019

 17.66 ms

Line 27-35 in File SafeMath.sol

```
27  /*@CTK "SafeMath add"
28     @tag spec
29     @tag is_pure
30     @post (a + b < a || a + b < b) == __reverted
31     @post !__reverted -> __return == a + b
32     @post !__reverted -> !__has_overflow
33     @post !__reverted -> !__has_assertion_failure
34     @post !(__has_buf_overflow)
35  */
```

Line 36-41 in File SafeMath.sol


```
36  function add(uint256 a, uint256 b) internal pure returns (uint256) {
37      uint256 c = a + b;
38      require(c >= a, "SafeMath: addition overflow");
39
40      return c;
41  }
```

 The code meets the specification.

Formal Verification Request 2

SafeMath sub

 03, Jul 2019

 15.48 ms

Line 52-60 in File SafeMath.sol

```
52  /*@CTK "SafeMath sub"
53     @tag spec
54     @tag is_pure
55     @post (b > a) == __reverted
56     @post !__reverted -> __return == a - b
57     @post !__reverted -> !__has_overflow
58     @post !__reverted -> !__has_assertion_failure
59     @post !(__has_buf_overflow)
60  */
```

Line 61-66 in File SafeMath.sol


```
61  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
62      require(b <= a, "SafeMath: subtraction overflow");
63      uint256 c = a - b;
64
65      return c;
66  }
```

 The code meets the specification.

Formal Verification Request 3

SafeMath mul zero

 03, Jul 2019

 19.2 ms

Line 77-82 in File SafeMath.sol

```
77  /*@CTK "SafeMath mul zero"
78      @tag spec
79      @tag is_pure
80      @pre (a == 0)
81      @post __return == 0
82  */
```

Line 93-105 in File SafeMath.sol


```
93  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
94      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
95      // benefit is lost if 'b' is also tested.
96      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
97      if (a == 0) {
98          return 0;
99      }
100
101      uint256 c = a * b;
102      require(c / a == b, "SafeMath: multiplication overflow");
103
104      return c;
105  }
```

 The code meets the specification.

Formal Verification Request 4

SafeMath mul nonzero

 03, Jul 2019

 289.32 ms

Line 83-92 in File SafeMath.sol

```
83  /*@CTK "SafeMath mul nonzero"
84      @tag spec
85      @tag is_pure
86      @pre (a != 0)
87      @post (a * b / a != b) == __reverted
88      @post !__reverted -> __return == a * b
89      @post !__reverted -> !__has_overflow
90      @post !__reverted -> !__has_assertion_failure
91      @post !(__has_buf_overflow)
92  */
```

Line 93-105 in File SafeMath.sol

```
93  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
94      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
95      // benefit is lost if 'b' is also tested.
```



```
96      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
97      if (a == 0) {
98          return 0;
99      }
100
101      uint256 c = a * b;
102      require(c / a == b, "SafeMath: multiplication overflow");
103
104      return c;
105  }
```

✓ The code meets the specification.

Formal Verification Request 5

SafeMath div



03, Jul 2019



13.18 ms

Line 118-126 in File SafeMath.sol

```
118  /*@CTK "SafeMath div"
119      @tag spec
120      @tag is_pure
121      @post (b == 0) == __reverted
122      @post !__reverted -> __return == a / b
123      @post !__reverted -> !__has_overflow
124      @post !__reverted -> !__has_assertion_failure
125      @post !(__has_buf_overflow)
126  */
```

Line 127-134 in File SafeMath.sol

```
127  function div(uint256 a, uint256 b) internal pure returns (uint256) {
128      // Solidity only automatically asserts when dividing by 0
129      require(b > 0, "SafeMath: division by zero");
130      uint256 c = a / b;
131      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
132
133      return c;
134  }
```

✓ The code meets the specification.

Formal Verification Request 6

SafeMath mod



03, Jul 2019



12.0 ms

Line 147-155 in File SafeMath.sol

```
147  /*@CTK "SafeMath mod"
148      @tag spec
```

```

149     @tag is_pure
150     @post (b == 0) == __reverted
151     @post !__reverted -> __return == a % b
152     @post !__reverted -> !__has_overflow
153     @post !__reverted -> !__has_assertion_failure
154     @post !(__has_buf_overflow)
155     */

```

Line 156-159 in File SafeMath.sol

```

156     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
157         require(b != 0, "SafeMath: modulo by zero");
158         return a % b;
159     }

```

✓ The code meets the specification.

Formal Verification Request 7

HMTOKEN

📅 03, Jul 2019

🕒 20.66 ms

Line 24-30 in File HMTOKEN.sol

```

24     /*@CTK HMTOKEN
25     @tag assume_completion
26     @post __post.name == _name
27     @post __post.decimals == _decimals
28     @post __post.symbol == _symbol
29     @post __post.balances[msg.sender] == __post.totalSupply
30     */

```

Line 31-37 in File HMTOKEN.sol

```

31     constructor(uint256 _totalSupply, string memory _name, uint8 _decimals, string
32         memory _symbol) public {
33         totalSupply = _totalSupply * (10 ** uint256(_decimals));
34         name = _name;
35         decimals = _decimals;
36         symbol = _symbol;
37         balances[msg.sender] = totalSupply;
38     }

```

✓ The code meets the specification.

Formal Verification Request 8

transfer

📅 03, Jul 2019

🕒 158.41 ms

Line 39-48 in File HMTOKEN.sol

```

39  /*@CTK transfer
40      @tag assume_completion
41      @pre msg.sender != _to
42      @post _to != address(0)
43      @post _to != address(this)
44      @post balances[msg.sender] >= _value
45      @post (balances[_to] + _value) >= balances[_to]
46      @post __post.balances[msg.sender] == balances[msg.sender] - _value
47      @post __post.balances[_to] == balances[_to] + _value
48  */

```

Line 49-53 in File HMTOKEN.sol

```

49  function transfer(address _to, uint256 _value) public returns (bool success) {
50      success = transferQuiet(_to, _value);
51      require(success, "Transfer didn't succeed");
52      return success;
53  }

```

✓ The code meets the specification.

Formal Verification Request 9

transferBulk

📅 03, Jul 2019

🕒 34.42 ms

Line 55-59 in File HMTOKEN.sol

```

55  /*@CTK transferBulk
56      @tag assume_completion
57      @pre _tos.length == _values.length
58      @pre _tos.length < BULK_MAX_COUNT
59  */

```

Line 60-92 in File HMTOKEN.sol

```

60  function transferBulk(address[] memory _tos, uint256[] memory _values, uint256
61      _txId) public returns (uint256 _bulkCount) {
62      require(_tos.length == _values.length, "Amount of recipients and values don't
63          match");
64      require(_tos.length < BULK_MAX_COUNT, "Too many recipients");
65
66      uint256 _bulkValue = 0;
67      /*@CTK transferBulk_bulkValue_sum
68          @inv j <= _tos.length
69          @inv _bulkValue >= _bulkValue__pre
70          @post !__should_return
71      */
72      for (uint256 j = 0; j < _tos.length; ++j) {
73          _bulkValue = _bulkValue.add(_values[j]);
74      }
75      require(_bulkValue < BULK_MAX_VALUE, "Bulk value too high");
76
77      _bulkCount = 0;
78      bool _success;
79      /*@CTK transferBulk_transfer

```

```

78     @inv i <= _tos.length
79     @inv _bulkCount >= _bulkCount__pre
80     @post !__should_return
81     */
82     for (uint256 i = 0; i < _tos.length; ++i) {
83         _success = transferQuiet(_tos[i], _values[i]);
84         if (_success) {
85             _bulkCount = _bulkCount.add(1);
86         } else {
87             emit BulkTransferFailure(_txId, _bulkCount);
88         }
89     }
90     emit BulkTransfer(_txId, _bulkCount);
91     return _bulkCount;
92 }

```

✓ The code meets the specification.

Formal Verification Request 10

transferFrom

03, Jul 2019

270.0 ms

Line 94-103 in File HMTOKEN.sol

```

94     /*@CTK transferFrom
95     @tag assume_completion
96     @pre _spender != _to
97     @post balances[_spender] >= _value && allowed[_spender][msg.sender] >= _value
98     @post _to != address(0)
99     @post __post.balances[_spender] == balances[_spender] - _value
100     @post __post.balances[_to] == balances[_to] + _value
101     @post (allowed[_spender][msg.sender] < MAX_UINT256) -> (__post.allowed[_spender]
102         [msg.sender] == allowed[_spender][msg.sender] - _value)
103     @post (allowed[_spender][msg.sender] >= MAX_UINT256) -> (__post.allowed[_spender]
104         [msg.sender] == allowed[_spender][msg.sender])
105     */

```

Line 104-118 in File HMTOKEN.sol

```

104     function transferFrom(address _spender, address _to, uint256 _value) public
105         returns (bool success) {
106         uint256 _allowance = allowed[_spender][msg.sender];
107         require(balances[_spender] >= _value && _allowance >= _value, "Spender balance
108             or allowance too low");
109         require(_to != address(0), "Can't send tokens to uninitialized address");
110
111         balances[_spender] = balances[_spender].sub(_value);
112         balances[_to] = balances[_to].add(_value);
113
114         if (_allowance < MAX_UINT256) {
115             allowed[_spender][msg.sender] = allowed[_spender][msg.sender].sub(_value);
116         }
117
118         emit Transfer(_spender, _to, _value);
119         return true;

```




118 }

✓ The code meets the specification.

Formal Verification Request 11

approve

03, Jul 2019

19.46 ms

Line 120-124 in File HMToken.sol

```

120  /*@CTK approve
121      @tag assume_completion
122      @post _spender != address(0)
123      @post __post.allowed[msg.sender][_spender] == _value
124  */

```

Line 125-131 in File HMToken.sol

```

125  function approve(address _spender, uint256 _value) public returns (bool success) {
126      require(_spender != address(0), "Token spender is an uninitialized address");
127
128      allowed[msg.sender][_spender] = _value;
129      emit Approval(msg.sender, _spender, _value); //solhint-disable-line indent, no-
        unused-vars
130      return true;
131  }

```

✓ The code meets the specification.

Formal Verification Request 12

approveBulk

03, Jul 2019

33.51 ms

Line 133-137 in File HMToken.sol

```

133  /*@CTK approveBulk
134      @tag assume_completion
135      @pre _spenders.length == _values.length
136      @pre _spenders.length < BULK_MAX_COUNT
137  */

```

Line 138-170 in File HMToken.sol

```

138  function approveBulk(address[] memory _spenders, uint256[] memory _values,
        uint256 _txId) public returns (uint256 _bulkCount) {
139      require(_spenders.length == _values.length, "Amount of spenders and values don'
        t match");
140      require(_spenders.length < BULK_MAX_COUNT, "Too many spenders");
141
142      uint256 _bulkValue = 0;
143      /*@CTK approveBulk_bulkValue_sum

```

```

144     @inv j <= _spenders.length
145     @inv _bulkValue >= _bulkValue__pre
146     @post !__should_return
147     */
148     for (uint256 j = 0; j < _spenders.length; ++j) {
149         _bulkValue = _bulkValue.add(_values[j]);
150     }
151     require(_bulkValue < BULK_MAX_VALUE, "Bulk value too high");
152
153     _bulkCount = 0;
154     bool _success;
155     /*@CTK approveBulk_approve
156     @inv i <= _spenders.length
157     @inv _bulkCount >= _bulkCount__pre
158     @post !__should_return
159     */
160     for (uint256 i = 0; i < _spenders.length; ++i) {
161         _success = increaseApproval(_spenders[i], _values[i]);
162         if (_success) {
163             _bulkCount = _bulkCount.add(1);
164         } else {
165             emit BulkApprovalFailure(_txId, _bulkCount);
166         }
167     }
168     emit BulkApproval(_txId, _bulkCount);
169     return _bulkCount;
170 }

```

✓ The code meets the specification.

Formal Verification Request 13

increaseApproval

📅 03, Jul 2019

🕒 304.54 ms

Line 172-178 in File HMTOKEN.sol

```

172     /*@CTK increaseApproval
173     @tag assume_completion
174     @pre allowed[msg.sender][_spender] + _delta < MAX_UINT256
175     @post _spender != address(0)
176     @post __post.allowed[msg.sender][_spender] ==
177         allowed[msg.sender][_spender] + _delta
178     */

```

Line 179-190 in File HMTOKEN.sol

```

179     function increaseApproval(address _spender, uint256 _delta) public returns (bool
180         success) {
181         require(_spender != address(0), "Token spender is an uninitialized address");
182
183         uint256 _oldValue = allowed[msg.sender][_spender];
184         if (_oldValue.add(_delta) < _oldValue || _oldValue.add(_delta) >= MAX_UINT256)
185             { // Truncate upon overflow.
186                 allowed[msg.sender][_spender] = MAX_UINT256.sub(1);
187             } else {

```



```

186         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_delta);
187     }
188     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
189     return true;
190 }

```

✓ The code meets the specification.

Formal Verification Request 14

decreaseApproval

📅 03, Jul 2019

🕒 90.77 ms

Line 192-200 in File HMTOKEN.sol

```

192  /*@CTK decreaseApproval
193     @tag assume_completion
194     @post _spender != address(0)
195     @post _delta > allowed[msg.sender][_spender] ->
196         __post.allowed[msg.sender][_spender] == 0
197     @post _delta <= allowed[msg.sender][_spender]
198         -> __post.allowed[msg.sender][_spender] ==
199             allowed[msg.sender][_spender] - _delta
200 */

```

Line 201-212 in File HMTOKEN.sol

```

201  function decreaseApproval(address _spender, uint256 _delta) public returns (bool
      success) {
202      require(_spender != address(0), "Token spender is an uninitialized address");
203
204      uint256 _oldValue = allowed[msg.sender][_spender];
205      if (_delta > _oldValue) { // Truncate upon overflow.
206          allowed[msg.sender][_spender] = 0;
207      } else {
208          allowed[msg.sender][_spender] = allowed[msg.sender][_spender].sub(_delta);
209      }
210      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
211      return true;
212  }

```

✓ The code meets the specification.

Formal Verification Request 15

allowance

📅 03, Jul 2019

🕒 4.49 ms

Line 214-216 in File HMTOKEN.sol

```

214  /*@CTK allowance
215     @post remaining == allowed[_owner][_spender]
216 */

```

Line 217-219 in File HMTOKEN.sol

```
217     function allowance(address _owner, address _spender) public view returns (uint256
        remaining) {
218         return allowed[_owner][_spender];
219     }
```

✓ The code meets the specification.

Formal Verification Request 16

balanceOf

📅 03, Jul 2019

🕒 4.51 ms

Line 221-223 in File HMTOKEN.sol

```
221     /*@CTK balanceOf
222         @post balance == __post.balances[_owner]
223     */
```

Line 224-226 in File HMTOKEN.sol

```
224     function balanceOf(address _owner) public view returns (uint256 balance) {
225         return balances[_owner];
226     }
```

✓ The code meets the specification.

Formal Verification Request 17

transferQuiet

📅 03, Jul 2019

🕒 48.93 ms

Line 229-238 in File HMTOKEN.sol

```
229     /*@CTK transferQuiet
230         @pre msg.sender != _to
231         @pre _to != address(0)
232         @pre _to != address(this)
233         @pre balances[msg.sender] >= _value
234         @pre (balances[_to] + _value) >= balances[_to]
235         @post __post.balances[msg.sender] == balances[msg.sender] - _value
236         @post __post.balances[_to] == balances[_to] + _value
237         @post success
238     */
```

Line 239-250 in File HMTOKEN.sol

```
239     function transferQuiet(address _to, uint256 _value) internal returns (bool success)
        {
240         if (_to == address(0)) return false; // Preclude burning tokens to
            uninitialized address.
241         if (_to == address(this)) return false; // Preclude sending tokens to the
            contract.
```

```

242     if (balances[msg.sender] < _value) return false; // Preclude transferring more
                than sender's balance.
243     if (balances[_to] + _value < balances[_to]) return false; // Handle overflow
                here in order to avoid reverts from SafeMath.
244
245     balances[msg.sender] = balances[msg.sender].sub(_value);
246     balances[_to] = balances[_to].add(_value);
247
248     emit Transfer(msg.sender, _to, _value);
249     return true;
250 }


```

✓ The code meets the specification.

Formal Verification Request 18

transferBulk_bulkValue_sum__Generated

 03, Jul 2019

 48.11 ms

(Loop) Line 65-69 in File HMToken.sol

```

65     /*@CTK transferBulk_bulkValue_sum
66         @inv j <= _tos.length
67         @inv _bulkValue >= _bulkValue__pre
68         @post !__should_return
69     */

```

(Loop) Line 65-72 in File HMToken.sol

```

65     /*@CTK transferBulk_bulkValue_sum
66         @inv j <= _tos.length
67         @inv _bulkValue >= _bulkValue__pre
68         @post !__should_return
69     */
70     for (uint256 j = 0; j < _tos.length; ++j) {
71         _bulkValue = _bulkValue.add(_values[j]);
72     }


```

✓ The code meets the specification.

Formal Verification Request 19

transferBulk_transfer__Generated

 03, Jul 2019

 159.61 ms

(Loop) Line 77-81 in File HMToken.sol

```

77     /*@CTK transferBulk_transfer
78         @inv i <= _tos.length
79         @inv _bulkCount >= _bulkCount__pre
80         @post !__should_return
81     */

```

(Loop) Line 77-89 in File HMTOKEN.sol

```

77      /*@CTK transferBulk_transfer
78         @inv i <= _tos.length
79         @inv _bulkCount >= _bulkCount__pre
80         @post !__should_return
81      */
82      for (uint256 i = 0; i < _tos.length; ++i) {
83          _success = transferQuiet(_tos[i], _values[i]);
84          if (_success) {
85              _bulkCount = _bulkCount.add(1);
86          } else {
87              emit BulkTransferFailure(_txId, _bulkCount);
88          }
89      }

```

✓ The code meets the specification.

Formal Verification Request 20

approveBulk_bulkValue_sum__Generated

📅 03, Jul 2019

🕒 42.13 ms

(Loop) Line 143-147 in File HMTOKEN.sol

```

143      /*@CTK approveBulk_bulkValue_sum
144         @inv j <= _spenders.length
145         @inv _bulkValue >= _bulkValue__pre
146         @post !__should_return
147      */

```

(Loop) Line 143-150 in File HMTOKEN.sol

```

143      /*@CTK approveBulk_bulkValue_sum
144         @inv j <= _spenders.length
145         @inv _bulkValue >= _bulkValue__pre
146         @post !__should_return
147      */
148      for (uint256 j = 0; j < _spenders.length; ++j) {
149          _bulkValue = _bulkValue.add(_values[j]);
150      }

```

✓ The code meets the specification.

Formal Verification Request 21

approveBulk_approve__Generated

📅 03, Jul 2019

🕒 146.76 ms

(Loop) Line 155-159 in File HMTOKEN.sol

```

155      /*@CTK approveBulk_approve
156      @inv i <= _spenders.length
157      @inv _bulkCount >= _bulkCount__pre
158      @post !__should_return
159      */

```

(Loop) Line 155-167 in File HMTOKEN.sol

```

155      /*@CTK approveBulk_approve
156      @inv i <= _spenders.length
157      @inv _bulkCount >= _bulkCount__pre
158      @post !__should_return
159      */
160      for (uint256 i = 0; i < _spenders.length; ++i) {
161          _success = increaseApproval(_spenders[i], _values[i]);
162          if (_success) {
163              _bulkCount = _bulkCount.add(1);
164          } else {
165              emit BulkApprovalFailure(_txId, _bulkCount);
166          }
167      }

```

✓ The code meets the specification.

Source Code with CertiK Labels

File SafeMath.sol

```

1  pragma solidity 0.5.9;
2
3
4  /**
5   * @dev Wrappers over Solidity's arithmetic operations with added overflow
6   * checks.
7   *
8   * Arithmetic operations in Solidity wrap on overflow. This can easily result
9   * in bugs, because programmers usually assume that an overflow raises an
10  * error, which is the standard behavior in high level programming languages.
11  * 'SafeMath' restores this intuition by reverting the transaction when an
12  * operation overflows.
13  *
14  * Using this library instead of the unchecked operations eliminates an entire
15  * class of bugs, so it's recommended to use it always.
16  */
17  library SafeMath {
18      /**
19       * @dev Returns the addition of two unsigned integers, reverting on
20       * overflow.
21       *
22       * Counterpart to Solidity's '+' operator.
23       *
24       * Requirements:
25       * - Addition cannot overflow.
26       */
27      /*@CTK "SafeMath add"
28       @tag spec
29       @tag is_pure
30       @post (a + b < a || a + b < b) == __reverted
31       @post !__reverted -> __return == a + b
32       @post !__reverted -> !__has_overflow
33       @post !__reverted -> !__has_assertion_failure
34       @post !(__has_buf_overflow)
35      */
36      function add(uint256 a, uint256 b) internal pure returns (uint256) {
37          uint256 c = a + b;
38          require(c >= a, "SafeMath: addition overflow");
39
40          return c;
41      }
42
43      /**
44       * @dev Returns the subtraction of two unsigned integers, reverting on
45       * overflow (when the result is negative).
46       *
47       * Counterpart to Solidity's '-' operator.
48       *
49       * Requirements:
50       * - Subtraction cannot overflow.
51       */
52      /*@CTK "SafeMath sub"
53       @tag spec
54       @tag is_pure

```



```

55     @post (b > a) == __reverted
56     @post !__reverted -> __return == a - b
57     @post !__reverted -> !__has_overflow
58     @post !__reverted -> !__has_assertion_failure
59     @post !(__has_buf_overflow)
60 */
61 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
62     require(b <= a, "SafeMath: subtraction overflow");
63     uint256 c = a - b;
64
65     return c;
66 }
67
68 /**
69  * @dev Returns the multiplication of two unsigned integers, reverting on
70  * overflow.
71  *
72  * Counterpart to Solidity's '*' operator.
73  *
74  * Requirements:
75  * - Multiplication cannot overflow.
76  */
77 /*@CTK "SafeMath mul zero"
78  @tag spec
79  @tag is_pure
80  @pre (a == 0)
81  @post __return == 0
82  */
83 /*@CTK "SafeMath mul nonzero"
84  @tag spec
85  @tag is_pure
86  @pre (a != 0)
87  @post (a * b / a != b) == __reverted
88  @post !__reverted -> __return == a * b
89  @post !__reverted -> !__has_overflow
90  @post !__reverted -> !__has_assertion_failure
91  @post !(__has_buf_overflow)
92  */
93 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
94     // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
95     // benefit is lost if 'b' is also tested.
96     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
97     if (a == 0) {
98         return 0;
99     }
100
101     uint256 c = a * b;
102     require(c / a == b, "SafeMath: multiplication overflow");
103
104     return c;
105 }
106
107 /**
108  * @dev Returns the integer division of two unsigned integers. Reverts on
109  * division by zero. The result is rounded towards zero.
110  *
111  * Counterpart to Solidity's '/' operator. Note: this function uses a
112  * 'revert' opcode (which leaves remaining gas untouched) while Solidity

```

```

113     * uses an invalid opcode to revert (consuming all remaining gas).
114     *
115     * Requirements:
116     * - The divisor cannot be zero.
117     */
118     /*@CTK "SafeMath div"
119     @tag spec
120     @tag is_pure
121     @post (b == 0) == __reverted
122     @post !__reverted -> __return == a / b
123     @post !__reverted -> !__has_overflow
124     @post !__reverted -> !__has_assertion_failure
125     @post !(__has_buf_overflow)
126     */
127     function div(uint256 a, uint256 b) internal pure returns (uint256) {
128         // Solidity only automatically asserts when dividing by 0
129         require(b > 0, "SafeMath: division by zero");
130         uint256 c = a / b;
131         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
132
133         return c;
134     }
135
136     /**
137     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
138         modulo),
139     * Reverts when dividing by zero.
140     *
141     * Counterpart to Solidity's '%' operator. This function uses a 'revert'
142     * opcode (which leaves remaining gas untouched) while Solidity uses an
143     * invalid opcode to revert (consuming all remaining gas).
144     *
145     * Requirements:
146     * - The divisor cannot be zero.
147     */
148     /*@CTK "SafeMath mod"
149     @tag spec
150     @tag is_pure
151     @post (b == 0) == __reverted
152     @post !__reverted -> __return == a % b
153     @post !__reverted -> !__has_overflow
154     @post !__reverted -> !__has_assertion_failure
155     @post !(__has_buf_overflow)
156     */
157     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
158         require(b != 0, "SafeMath: modulo by zero");
159         return a % b;
160     }

```

File HMTOKEN.sol

```

1 pragma solidity 0.5.9;
2
3 /*
4 Implements EIP20 token standard: https://github.com/ethereum/EIPs/issues/20
5 */
6
7 import "./SafeMath.sol";

```

```

8 import "./HMTOKENInterface.sol";
9
10
11 contract HMTOKEN is HMTOKENInterface {
12     using SafeMath for uint256;
13     uint256 private constant MAX_UINT256 = 2**256 - 1;
14     uint256 private constant BULK_MAX_VALUE = 1000000000 * (10 ** 18);
15     uint32 private constant BULK_MAX_COUNT = 100;
16
17     mapping (address => uint256) private balances;
18     mapping (address => mapping (address => uint256)) private allowed;
19
20     string public name;
21     uint8 public decimals;
22     string public symbol;
23
24     /*@CTK HMTOKEN
25     @tag assume_completion
26     @post __post.name == _name
27     @post __post.decimals == _decimals
28     @post __post.symbol == _symbol
29     @post __post.balances[msg.sender] == __post.totalSupply
30     */
31     constructor(uint256 _totalSupply, string memory _name, uint8 _decimals, string
        memory _symbol) public {
32         totalSupply = _totalSupply * (10 ** uint256(_decimals));
33         name = _name;
34         decimals = _decimals;
35         symbol = _symbol;
36         balances[msg.sender] = totalSupply;
37     }
38
39     /*@CTK transfer
40     @tag assume_completion
41     @pre msg.sender != _to
42     @post _to != address(0)
43     @post _to != address(this)
44     @post balances[msg.sender] >= _value
45     @post (balances[_to] + _value) >= balances[_to]
46     @post __post.balances[msg.sender] == balances[msg.sender] - _value
47     @post __post.balances[_to] == balances[_to] + _value
48     */
49     function transfer(address _to, uint256 _value) public returns (bool success) {
50         success = transferQuiet(_to, _value);
51         require(success, "Transfer didn't succeed");
52         return success;
53     }
54
55     /*@CTK transferBulk
56     @tag assume_completion
57     @pre _tos.length == _values.length
58     @pre _tos.length < BULK_MAX_COUNT
59     */
60     function transferBulk(address[] memory _tos, uint256[] memory _values, uint256
        _txId) public returns (uint256 _bulkCount) {
61         require(_tos.length == _values.length, "Amount of recipients and values don't
            match");
62         require(_tos.length < BULK_MAX_COUNT, "Too many recipients");

```

```

63
64     uint256 _bulkValue = 0;
65     /*@CTK transferBulk_bulkValue_sum
66         @inv j <= _tos.length
67         @inv _bulkValue >= _bulkValue__pre
68         @post !__should_return
69     */
70     for (uint256 j = 0; j < _tos.length; ++j) {
71         _bulkValue = _bulkValue.add(_values[j]);
72     }
73     require(_bulkValue < BULK_MAX_VALUE, "Bulk value too high");
74
75     _bulkCount = 0;
76     bool _success;
77     /*@CTK transferBulk_transfer
78         @inv i <= _tos.length
79         @inv _bulkCount >= _bulkCount__pre
80         @post !__should_return
81     */
82     for (uint256 i = 0; i < _tos.length; ++i) {
83         _success = transferQuiet(_tos[i], _values[i]);
84         if (_success) {
85             _bulkCount = _bulkCount.add(1);
86         } else {
87             emit BulkTransferFailure(_txId, _bulkCount);
88         }
89     }
90     emit BulkTransfer(_txId, _bulkCount);
91     return _bulkCount;
92 }
93
94 /*@CTK transferFrom
95     @tag assume_completion
96     @pre _spender != _to
97     @post balances[_spender] >= _value && allowed[_spender][msg.sender] >= _value
98     @post _to != address(0)
99     @post __post.balances[_spender] == balances[_spender] - _value
100    @post __post.balances[_to] == balances[_to] + _value
101    @post (allowed[_spender][msg.sender] < MAX_UINT256) -> (__post.allowed[_spender]
102        [msg.sender] == allowed[_spender][msg.sender] - _value)
103    @post (allowed[_spender][msg.sender] >= MAX_UINT256) -> (__post.allowed[_spender]
104        [msg.sender] == allowed[_spender][msg.sender])
105 */
106 function transferFrom(address _spender, address _to, uint256 _value) public
107     returns (bool success) {
108     uint256 _allowance = allowed[_spender][msg.sender];
109     require(balances[_spender] >= _value && _allowance >= _value, "Spender balance
110         or allowance too low");
111     require(_to != address(0), "Can't send tokens to uninitialized address");
112
113     balances[_spender] = balances[_spender].sub(_value);
114     balances[_to] = balances[_to].add(_value);
115
116     if (_allowance < MAX_UINT256) {
117         allowed[_spender][msg.sender] = allowed[_spender][msg.sender].sub(_value);
118     }
119
120     emit Transfer(_spender, _to, _value);

```

```

117     return true;
118 }
119
120 /*@CTK approve
121   @tag assume_completion
122   @post _spender != address(0)
123   @post __post.allowed[msg.sender][_spender] == _value
124 */
125 function approve(address _spender, uint256 _value) public returns (bool success) {
126     require(_spender != address(0), "Token spender is an uninitialized address");
127
128     allowed[msg.sender][_spender] = _value;
129     emit Approval(msg.sender, _spender, _value); //solhint-disable-line indent, no-
        unused-vars
130     return true;
131 }
132
133 /*@CTK approveBulk
134   @tag assume_completion
135   @pre _spenders.length == _values.length
136   @pre _spenders.length < BULK_MAX_COUNT
137 */
138 function approveBulk(address[] memory _spenders, uint256[] memory _values,
139     uint256 _txId) public returns (uint256 _bulkCount) {
140     require(_spenders.length == _values.length, "Amount of spenders and values don'
        t match");
141     require(_spenders.length < BULK_MAX_COUNT, "Too many spenders");
142
143     uint256 _bulkValue = 0;
144     /*@CTK approveBulk_bulkValue_sum
145       @inv j <= _spenders.length
146       @inv _bulkValue >= _bulkValue__pre
147       @post !__should_return
148     */
149     for (uint256 j = 0; j < _spenders.length; ++j) {
150         _bulkValue = _bulkValue.add(_values[j]);
151     }
152     require(_bulkValue < BULK_MAX_VALUE, "Bulk value too high");
153
154     _bulkCount = 0;
155     bool _success;
156     /*@CTK approveBulk_approve
157       @inv i <= _spenders.length
158       @inv _bulkCount >= _bulkCount__pre
159       @post !__should_return
160     */
161     for (uint256 i = 0; i < _spenders.length; ++i) {
162         _success = increaseApproval(_spenders[i], _values[i]);
163         if (_success) {
164             _bulkCount = _bulkCount.add(1);
165         } else {
166             emit BulkApprovalFailure(_txId, _bulkCount);
167         }
168     }
169     emit BulkApproval(_txId, _bulkCount);
170     return _bulkCount;
171 }

```

```

172  /*@CTK increaseApproval
173      @tag assume_completion
174      @pre allowed[msg.sender][_spender] + _delta < MAX_UINT256
175      @post _spender != address(0)
176      @post __post.allowed[msg.sender][_spender] ==
177          allowed[msg.sender][_spender] + _delta
178  */
179  function increaseApproval(address _spender, uint256 _delta) public returns (bool
    success) {
180      require(_spender != address(0), "Token spender is an uninitialized address");
181
182      uint256 _oldValue = allowed[msg.sender][_spender];
183      if (_oldValue.add(_delta) < _oldValue || _oldValue.add(_delta) >= MAX_UINT256)
184          { // Truncate upon overflow.
185              allowed[msg.sender][_spender] = MAX_UINT256.sub(1);
186          } else {
187              allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_delta);
188          }
189      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
190      return true;
191  }
192
193  /*@CTK decreaseApproval
194      @tag assume_completion
195      @post _spender != address(0)
196      @post _delta > allowed[msg.sender][_spender] ->
197          __post.allowed[msg.sender][_spender] == 0
198      @post _delta <= allowed[msg.sender][_spender]
199          -> __post.allowed[msg.sender][_spender] ==
200              allowed[msg.sender][_spender] - _delta
201  */
202  function decreaseApproval(address _spender, uint256 _delta) public returns (bool
    success) {
203      require(_spender != address(0), "Token spender is an uninitialized address");
204
205      uint256 _oldValue = allowed[msg.sender][_spender];
206      if (_delta > _oldValue) { // Truncate upon overflow.
207          allowed[msg.sender][_spender] = 0;
208      } else {
209          allowed[msg.sender][_spender] = allowed[msg.sender][_spender].sub(_delta);
210      }
211      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
212      return true;
213  }
214
215  /*@CTK allowance
216      @post remaining == allowed[_owner][_spender]
217  */
218  function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
219      return allowed[_owner][_spender];
220  }
221
222  /*@CTK balanceOf
223      @post balance == __post.balances[_owner]
224  */
225  function balanceOf(address _owner) public view returns (uint256 balance) {
    return balances[_owner];

```

```

226     }
227
228     // Like transfer, but fails quietly.
229     /*@CTK transferQuiet
230     @pre msg.sender != _to
231     @pre _to != address(0)
232     @pre _to != address(this)
233     @pre balances[msg.sender] >= _value
234     @pre (balances[_to] + _value) >= balances[_to]
235     @post __post.balances[msg.sender] == balances[msg.sender] - _value
236     @post __post.balances[_to] == balances[_to] + _value
237     @post success
238     */
239     function transferQuiet(address _to, uint256 _value) internal returns (bool success)
240     {
241         if (_to == address(0)) return false; // Preclude burning tokens to
242             uninitialized address.
243         if (_to == address(this)) return false; // Preclude sending tokens to the
244             contract.
245         if (balances[msg.sender] < _value) return false; // Preclude transferring more
246             than sender's balance.
247         if (balances[_to] + _value < balances[_to]) return false; // Handle overflow
248             here in order to avoid reverts from SafeMath.
249
250         balances[msg.sender] = balances[msg.sender].sub(_value);
251         balances[_to] = balances[_to].add(_value);
252
253         emit Transfer(msg.sender, _to, _value);
254         return true;
255     }
256 }

```