CertiK Audit Report
For ez365

Request Date: 2019-07-06
Revision Date: 2019-07-09
Platform Name: Ethereum

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 ez365(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/

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by ez365. This audit was conducted to discover issues and vulnerabilities in the source code of ez365'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 issue 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 conditions, 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 vulnerabilies, but no concern found yet.

Testing Summary

WARNING

CERTIK *identified some potential security flaws in this contract and also provided corresponding solutions.*

Jul 09, 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.	4	
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

Missing returns of several functions in `EZ365Token`:

- `transfer`: Change `super.transfer(_to,_value);` to `return super.transfer(_to,_value);`
- `transferFrom`: Change `super.transferFrom(_from, _to, _value);` to `return super.transferFrom(_from, _to, _value);`
- `increaseAllowance`: Change `super.increaseAllowance(_spender, _addedValue);` to `return super.increaseAllowance(_spender, _addedValue);`
- `decreaseAllowance`: Change `super.decreaseAllowance(_spender, _subtractedValue);` to `return super.decreaseAllowance(_spender, _subtractedValue);`

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **ez365.sol** `033d9c1c43a8f0d9f276fb87087ad2b7ea2f3f72e63ab749c61e32543929c16d`

Summary

CertiK was chosen by ez365 to audit the design and implementation of its soon to be released smart contract. To ensure comprehensive protection,the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 5 in File ez365.sol

```
5 pragma solidity ^0.5.2;
```

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

TIMESTAMP_DEPENDENCY

Line 597 in File ez365.sol

```
597             require(block.timestamp >= _releaseTime);
```

❗ ”block.timestamp” can be influenced by minors to some degree

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

Ownable

📅 09, Jul 2019
⏱ 18.51 ms

Line 20-22 in File ez365.sol

```
20    /*@CTK Ownable
21      @post __post._owner == msg.sender
22     */
```

Line 23-26 in File ez365.sol

```
23    constructor () internal {
24        _owner = msg.sender;
25        emit OwnershipTransferred(address(0), _owner);
26    }
```

✅ The code meets the specification.

Formal Verification Request 2

owner

📅 09, Jul 2019
⏱ 22.38 ms

Line 31-33 in File ez365.sol

```
31    /*@CTK owner
32      @post __return == _owner
33     */
```

Line 34-36 in File ez365.sol

```
34    function owner() public view returns (address) {
35        return _owner;
36    }
```

✅ The code meets the specification.

Formal Verification Request 3

isOwner

📅 09, Jul 2019
⏱ 18.56 ms

Line 49-51 in File ez365.sol

```
49    /*@CTK isOwner
50      @post __return == (msg.sender == _owner)
51     */
```

Line 52-54 in File ez365.sol

```
52      function isOwner() public view returns (bool) {
53          return msg.sender == _owner;
54      }
```

The code meets the specification.

Formal Verification Request 4

renounceOwnership

09, Jul 2019
104.68 ms

Line 63-67 in File ez365.sol

```
63      /*@CTK renounceOwnership
64        @tag assume_completion
65        @post _owner == msg.sender
66        @post __post._owner == address(0)
67       */
```

Line 68-71 in File ez365.sol

```
68      function renounceOwnership() public onlyOwner {
69          emit OwnershipTransferred(_owner, address(0));
70          _owner = address(0);
71      }
```

The code meets the specification.

Formal Verification Request 5

transferOwnership

09, Jul 2019
207.33 ms

Line 77-80 in File ez365.sol

```
77      /*@CTK transferOwnership
78        @tag assume_completion
79        @post _owner == msg.sender
80       */
```

Line 81-83 in File ez365.sol

```
81      function transferOwnership(address newOwner) public onlyOwner {
82          _transferOwnership(newOwner);
83      }
```

The code meets the specification.

Formal Verification Request 6

_transferOwnership

09, Jul 2019
2.6 ms

Line 89-93 in File ez365.sol

```
89    /*@CTK _transferOwnership
90      @tag assume_completion
91      @post newOwner != address(0)
92      @post __post._owner == newOwner
93     */
```

Line 94-98 in File ez365.sol

```
94    function _transferOwnership(address newOwner) internal {
95        require(newOwner != address(0));
96        emit OwnershipTransferred(_owner, newOwner);
97        _owner = newOwner;
98    }
```

The code meets the specification.

Formal Verification Request 7

SafeMath mul

09, Jul 2019
854.1 ms

Line 108-114 in File ez365.sol

```
108   /*@CTK "SafeMath mul"
109     @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
110     @post !__reverted -> __return == a * b
111     @post !__reverted == !__has_overflow
112     @post !(__has_buf_overflow)
113     @post !(__has_assertion_failure)
114    */
```

Line 115-127 in File ez365.sol

```
115   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
116       // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
117       // benefit is lost if 'b' is also tested.
118       // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
119       if (a == 0) {
120           return 0;
121       }
122
123       uint256 c = a * b;
124       require(c / a == b);
125
126       return c;
127   }
```

The code meets the specification.

Formal Verification Request 8

SafeMath div

09, Jul 2019
58.43 ms

Line 132-138 in File ez365.sol

```
132     /*@CTK "SafeMath div"
133       @post b != 0 -> !__reverted
134       @post !__reverted -> __return == a / b
135       @post !__reverted -> !__has_overflow
136       @post !(__has_buf_overflow)
137       @post !(__has_assertion_failure)
138      */
```

Line 139-146 in File ez365.sol

```
139     function div(uint256 a, uint256 b) internal pure returns (uint256) {
140         // Solidity only automatically asserts when dividing by 0
141         require(b > 0);
142         uint256 c = a / b;
143         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
144 
145         return c;
146     }
```

The code meets the specification.

Formal Verification Request 9

SafeMath sub

09, Jul 2019
46.74 ms

Line 151-157 in File ez365.sol

```
151     /*@CTK "SafeMath sub"
152       @post (a < b) == __reverted
153       @post !__reverted -> __return == a - b
154       @post !__reverted -> !__has_overflow
155       @post !(__has_buf_overflow)
156       @post !(__has_assertion_failure)
157      */
```

Line 158-163 in File ez365.sol

```
158     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
159         require(b <= a);
160         uint256 c = a - b;
161 
162         return c;
163     }
```

The code meets the specification.

Formal Verification Request 10

SafeMath add

09, Jul 2019
70.34 ms

Line 168-174 in File ez365.sol

```
168   /*@CTK "SafeMath add"
169     @post (a + b < a || a + b < b) == __reverted
170     @post !__reverted -> __return == a + b
171     @post !__reverted -> !__has_overflow
172     @post !(__has_buf_overflow)
173     @post !(__has_assertion_failure)
174    */
```

Line 175-180 in File ez365.sol

```
175   function add(uint256 a, uint256 b) internal pure returns (uint256) {
176       uint256 c = a + b;
177       require(c >= a);
178
179       return c;
180   }
```

The code meets the specification.

Formal Verification Request 11

SafeMath mod

09, Jul 2019
43.95 ms

Line 186-192 in File ez365.sol

```
186   /*@CTK "SafeMath mod"
187     @post b != 0 -> !__reverted
188     @post !__reverted -> __return == a % b
189     @post !__reverted -> !__has_overflow
190     @post !(__has_buf_overflow)
191     @post !(__has_assertion_failure)
192    */
```

Line 193-196 in File ez365.sol

```
193   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
194       require(b != 0);
195       return a % b;
196   }
```

The code meets the specification.

Formal Verification Request 12

If method completes, integer overflow would not happen.

09, Jul 2019
13.51 ms

Line 244 in File ez365.sol

```
244     //@CTK NO_OVERFLOW
```

Line 250-252 in File ez365.sol

```
250     function totalSupply() public view returns (uint256) {
251         return _totalSupply;
252     }
```

The code meets the specification.

Formal Verification Request 13

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

09, Jul 2019
0.83 ms

Line 245 in File ez365.sol

```
245     //@CTK NO_BUF_OVERFLOW
```

Line 250-252 in File ez365.sol

```
250     function totalSupply() public view returns (uint256) {
251         return _totalSupply;
252     }
```

The code meets the specification.

Formal Verification Request 14

Method will not encounter an assertion failure.

09, Jul 2019
0.72 ms

Line 246 in File ez365.sol

```
246     //@CTK NO_ASF
```

Line 250-252 in File ez365.sol

```
250     function totalSupply() public view returns (uint256) {
251         return _totalSupply;
252     }
```

The code meets the specification.

Formal Verification Request 15

totalSupply correctness

09, Jul 2019
0.78 ms

Line 247-249 in File ez365.sol

```
247     /*@CTK "totalSupply correctness"
248       @post __return == _totalSupply
249      */
```

Line 250-252 in File ez365.sol

```
250     function totalSupply() public view returns (uint256) {
251         return _totalSupply;
252     }
```

The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.

09, Jul 2019
15.04 ms

Line 259 in File ez365.sol

```
259     //@CTK NO_OVERFLOW
```

Line 265-267 in File ez365.sol

```
265     function balanceOf(address owner) public view returns (uint256) {
266         return _balances[owner];
267     }
```

The code meets the specification.

Formal Verification Request 17

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

09, Jul 2019
0.83 ms

Line 260 in File ez365.sol

```
260     //@CTK NO_BUF_OVERFLOW
```

Line 265-267 in File ez365.sol

```
265     function balanceOf(address owner) public view returns (uint256) {
266         return _balances[owner];
267     }
```

The code meets the specification.

Formal Verification Request 18

Method will not encounter an assertion failure.

09, Jul 2019
0.77 ms

Line 261 in File ez365.sol

```
261     //@CTK NO_ASF
```

Line 265-267 in File ez365.sol

```
265     function balanceOf(address owner) public view returns (uint256) {
266         return _balances[owner];
267     }
```

The code meets the specification.

Formal Verification Request 19

balanceOf correctness

09, Jul 2019
0.78 ms

Line 262-264 in File ez365.sol

```
262     /*@CTK "balanceOf correctness"
263       @post __return == _balances[owner]
264      */
```

Line 265-267 in File ez365.sol

```
265     function balanceOf(address owner) public view returns (uint256) {
266         return _balances[owner];
267     }
```

The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

09, Jul 2019
24.47 ms

Line 275 in File ez365.sol

```
275     //@CTK NO_OVERFLOW
```

Line 281-283 in File ez365.sol

```
281     function allowance(address owner, address spender) public view returns (uint256) {
282         return _allowed[owner][spender];
283     }
```

The code meets the specification.

Formal Verification Request 21

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

09, Jul 2019
0.79 ms

Line 276 in File ez365.sol

```
276     //@CTK NO_BUF_OVERFLOW
```

Line 281-283 in File ez365.sol

```
281     function allowance(address owner, address spender) public view returns (uint256) {
282         return _allowed[owner][spender];
283     }
```

The code meets the specification.

Formal Verification Request 22

Method will not encounter an assertion failure.

09, Jul 2019
0.8 ms

Line 277 in File ez365.sol

```
277     //@CTK NO_ASF
```

Line 281-283 in File ez365.sol

```
281     function allowance(address owner, address spender) public view returns (uint256) {
282         return _allowed[owner][spender];
283     }
```

The code meets the specification.

Formal Verification Request 23

allowance correctness

09, Jul 2019
1.2 ms

Line 278-280 in File ez365.sol

```
278     /*@CTK "allowance correctness"
279       @post __return == _allowed[owner][spender]
280      */
```

Line 281-283 in File ez365.sol

```
281     function allowance(address owner, address spender) public view returns (uint256) {
282         return _allowed[owner][spender];
283     }
```

The code meets the specification.

Formal Verification Request 24

If method completes, integer overflow would not happen.

09, Jul 2019
476.01 ms

Line 290 in File ez365.sol

```
290     //@CTK NO_OVERFLOW
```

Line 302-305 in File ez365.sol

```
302     function transfer(address to, uint256 value) public returns (bool) {
303         _transfer(msg.sender, to, value);
304         return true;
305     }
```

The code meets the specification.

Formal Verification Request 25

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

09, Jul 2019
14.62 ms

Line 291 in File ez365.sol

```
291     //@CTK NO_BUF_OVERFLOW
```

Line 302-305 in File ez365.sol

```
302     function transfer(address to, uint256 value) public returns (bool) {
303         _transfer(msg.sender, to, value);
304         return true;
305     }
```

The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

09, Jul 2019
12.77 ms

Line 292 in File ez365.sol

```
292     //@CTK NO_ASF
```

Line 302-305 in File ez365.sol

```
302     function transfer(address to, uint256 value) public returns (bool) {
303         _transfer(msg.sender, to, value);
304         return true;
305     }
```

The code meets the specification.

Formal Verification Request 27

transfer correctness

09, Jul 2019
163.32 ms

Line 293-301 in File ez365.sol

```
293     /*@CTK "transfer correctness"
294       @tag assume_completion
295       @post to != 0x0
296       @post value <= _balances[msg.sender]
297       @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - value
298       @post to != msg.sender -> __post._balances[to] == _balances[to] + value
299       @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
300       @post __return == true
301      */
```

Line 302-305 in File ez365.sol

```
302     function transfer(address to, uint256 value) public returns (bool) {
303         _transfer(msg.sender, to, value);
304         return true;
305     }
```

The code meets the specification.

Formal Verification Request 28

If method completes, integer overflow would not happen.

09, Jul 2019
183.23 ms

Line 316 in File ez365.sol

```
316     //@CTK NO_OVERFLOW
```

Line 324-327 in File ez365.sol

```
324     function approve(address spender, uint256 value) public returns (bool) {
325         _approve(msg.sender, spender, value);
326         return true;
327     }
```

The code meets the specification.

Formal Verification Request 29

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

09, Jul 2019
1.51 ms

Line 317 in File ez365.sol

```
317     //@CTK NO_BUF_OVERFLOW
```

Line 324-327 in File ez365.sol

```
324     function approve(address spender, uint256 value) public returns (bool) {
325         _approve(msg.sender, spender, value);
326         return true;
327     }
```

The code meets the specification.

Formal Verification Request 30

Method will not encounter an assertion failure.

09, Jul 2019
1.58 ms

Line 318 in File ez365.sol

```
318     //@CTK NO_ASF
```

Line 324-327 in File ez365.sol

```
324     function approve(address spender, uint256 value) public returns (bool) {
325         _approve(msg.sender, spender, value);
326         return true;
327     }
```

The code meets the specification.

Formal Verification Request 31

approve correctness

09, Jul 2019
7.91 ms

Line 319-323 in File ez365.sol

```
319     /*@CTK "approve correctness"
320       @pre msg.sender != 0x0
321       @post spender == 0x0 -> __reverted
322       @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
323      */
```

Line 324-327 in File ez365.sol

```
324     function approve(address spender, uint256 value) public returns (bool) {
325         _approve(msg.sender, spender, value);
326         return true;
327     }
```

The code meets the specification.

Formal Verification Request 32

If method completes, integer overflow would not happen.

09, Jul 2019
442.37 ms

Line 337 in File ez365.sol

```
337     //@CTK NO_OVERFLOW
```

Line 350-354 in File ez365.sol

```
350     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
351         _transfer(from, to, value);
352         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
353         return true;
354     }
```

The code meets the specification.

Formal Verification Request 33

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

09, Jul 2019
28.02 ms

Line 338 in File ez365.sol

```
338     //@CTK NO_BUF_OVERFLOW
```

Line 350-354 in File ez365.sol

```
350     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
351         _transfer(from, to, value);
352         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
353         return true;
354     }
```

The code meets the specification.

Formal Verification Request 34

Method will not encounter an assertion failure.

09, Jul 2019
18.61 ms

Line 339 in File ez365.sol

```
339     //@CTK NO_ASF
```

Line 350-354 in File ez365.sol

```
350     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
351         _transfer(from, to, value);
352         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
353         return true;
354     }
```

The code meets the specification.

Formal Verification Request 35

transferFrom correctness

09, Jul 2019
662.77 ms

Line 340-349 in File ez365.sol

```
340     /*@CTK "transferFrom correctness"
341       @tag assume_completion
342       @post to != 0x0
343       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
344       @post to != from -> __post._balances[from] == _balances[from] - value
345       @post to != from -> __post._balances[to] == _balances[to] + value
346       @post to == from -> __post._balances[from] == _balances[from]
347       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
348       @post __return == true
349      */
```

Line 350-354 in File ez365.sol

```
350     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
351         _transfer(from, to, value);
352         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
353         return true;
354     }
```

The code meets the specification.

Formal Verification Request 36

If method completes, integer overflow would not happen.

09, Jul 2019
205.76 ms

Line 366 in File ez365.sol

```
366     //@CTK NO_OVERFLOW
```

Line 375-378 in File ez365.sol

```
375     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
376         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
377         return true;
378     }
```

The code meets the specification.

Formal Verification Request 37

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

09, Jul 2019
2.19 ms

Line 367 in File ez365.sol

```
367     //@CTK NO_BUF_OVERFLOW
```

Line 375-378 in File ez365.sol

```
375     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
376         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
377         return true;
378     }
```

The code meets the specification.

Formal Verification Request 38

Method will not encounter an assertion failure.

09, Jul 2019
2.88 ms

Line 368 in File ez365.sol

```
368     //@CTK NO_ASF
```

Line 375-378 in File ez365.sol

```
375     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
376         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
377         return true;
378     }
```

The code meets the specification.

Formal Verification Request 39

increaseAllowance correctness

09, Jul 2019
7.98 ms

Line 369-374 in File ez365.sol

```
369   /*@CTK "increaseAllowance correctness"
370     @tag assume_completion
371     @post spender != 0x0
372     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
            addedValue
373     @post __return == true
374   */
```

Line 375-378 in File ez365.sol

```
375   function increaseAllowance(address spender, uint256 addedValue) public returns (
          bool) {
376       _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
377       return true;
378   }
```

The code meets the specification.

Formal Verification Request 40

If method completes, integer overflow would not happen.

09, Jul 2019
195.2 ms

Line 390 in File ez365.sol

```
390   //@CTK NO_OVERFLOW
```

Line 399-402 in File ez365.sol

```
399   function decreaseAllowance(address spender, uint256 subtractedValue) public
          returns (bool) {
400       _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
              ));
401       return true;
402   }
```

The code meets the specification.

Formal Verification Request 41

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

09, Jul 2019
3.77 ms

Line 391 in File ez365.sol

```
391   //@CTK NO_BUF_OVERFLOW
```

Line 399-402 in File ez365.sol

```
399   function decreaseAllowance(address spender, uint256 subtractedValue) public
          returns (bool) {
400       _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
              ));
401       return true;
402   }
```

The code meets the specification.

Formal Verification Request 42

Method will not encounter an assertion failure.

09, Jul 2019
4.45 ms

Line 392 in File ez365.sol

```
392     //@CTK NO_ASF
```

Line 399-402 in File ez365.sol

```
399     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
400         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
401         return true;
402     }
```

The code meets the specification.

Formal Verification Request 43

decreaseAllowance correctness

09, Jul 2019
15.16 ms

Line 393-398 in File ez365.sol

```
393     /*@CTK "decreaseAllowance correctness"
394       @tag assume_completion
395       @post spender != 0x0
396       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
397       @post __return == true
398      */
```

Line 399-402 in File ez365.sol

```
399     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
400         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
                ));
401         return true;
402     }
```

The code meets the specification.

Formal Verification Request 44

If method completes, integer overflow would not happen.

09, Jul 2019
225.38 ms

Line 425 in File ez365.sol

```
425     //@CTK NO_OVERFLOW
```

Line 434-440 in File ez365.sol

```
434     function _mint(address account, uint256 value) internal {
435         require(account != address(0));
436 
437         _totalSupply = _totalSupply.add(value);
438         _balances[account] = _balances[account].add(value);
439         emit Transfer(address(0), account, value);
440     }
```

The code meets the specification.

Formal Verification Request 45

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

09, Jul 2019
8.63 ms

Line 426 in File ez365.sol

```
426     //@CTK NO_BUF_OVERFLOW
```

Line 434-440 in File ez365.sol

```
434     function _mint(address account, uint256 value) internal {
435         require(account != address(0));
436 
437         _totalSupply = _totalSupply.add(value);
438         _balances[account] = _balances[account].add(value);
439         emit Transfer(address(0), account, value);
440     }
```

The code meets the specification.

Formal Verification Request 46

Method will not encounter an assertion failure.

09, Jul 2019
8.86 ms

Line 427 in File ez365.sol

```
427     //@CTK NO_ASF
```

Line 434-440 in File ez365.sol

```
434     function _mint(address account, uint256 value) internal {
435         require(account != address(0));
436 
437         _totalSupply = _totalSupply.add(value);
438         _balances[account] = _balances[account].add(value);
439         emit Transfer(address(0), account, value);
440     }
```

The code meets the specification.

Formal Verification Request 47

_mint correctness

09, Jul 2019
92.66 ms

Line 428-433 in File ez365.sol

```
428     /*@CTK "_mint correctness"
429       @tag assume_completion
430       @post account != 0x0
431       @post __post._balances[account] == _balances[account] + value
432       @post __post._totalSupply == _totalSupply + value
433      */
```

Line 434-440 in File ez365.sol

```
434     function _mint(address account, uint256 value) internal {
435         require(account != address(0));
436 
437         _totalSupply = _totalSupply.add(value);
438         _balances[account] = _balances[account].add(value);
439         emit Transfer(address(0), account, value);
440     }
```

The code meets the specification.

Formal Verification Request 48

If method completes, integer overflow would not happen.

09, Jul 2019
21.95 ms

Line 500 in File ez365.sol

```
500     //@CTK NO_OVERFLOW
```

Line 506-508 in File ez365.sol

```
506     function name() public pure returns (string memory) {
507         return _name;
508     }
```

The code meets the specification.

Formal Verification Request 49

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

09, Jul 2019
0.78 ms

Line 501 in File ez365.sol

```
501     //@CTK NO_BUF_OVERFLOW
```

Line 506-508 in File ez365.sol

```
506     function name() public pure returns (string memory) {
507         return _name;
508     }
```

The code meets the specification.

Formal Verification Request 50

Method will not encounter an assertion failure.

09, Jul 2019
0.78 ms

Line 502 in File ez365.sol

```
502     //@CTK NO_ASF
```

Line 506-508 in File ez365.sol

```
506     function name() public pure returns (string memory) {
507         return _name;
508     }
```

The code meets the specification.

Formal Verification Request 51

ERC20Detailed name correctness

09, Jul 2019
0.82 ms

Line 503-505 in File ez365.sol

```
503     /*@CTK "ERC20Detailed name correctness"
504       @post __return == _name
505      */
```

Line 506-508 in File ez365.sol

```
506     function name() public pure returns (string memory) {
507         return _name;
508     }
```

The code meets the specification.

Formal Verification Request 52

If method completes, integer overflow would not happen.

09, Jul 2019
19.13 ms

Line 513 in File ez365.sol

```
513     //@CTK NO_OVERFLOW
```

Line 519-521 in File ez365.sol

```
519     function symbol() public pure returns (string memory) {
520         return _symbol;
521     }
```

The code meets the specification.

Formal Verification Request 53

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

09, Jul 2019
0.95 ms

Line 514 in File ez365.sol

```
514     //@CTK NO_BUF_OVERFLOW
```

Line 519-521 in File ez365.sol

```
519     function symbol() public pure returns (string memory) {
520         return _symbol;
521     }
```

The code meets the specification.

Formal Verification Request 54

Method will not encounter an assertion failure.

09, Jul 2019
0.94 ms

Line 515 in File ez365.sol

```
515     //@CTK NO_ASF
```

Line 519-521 in File ez365.sol

```
519     function symbol() public pure returns (string memory) {
520         return _symbol;
521     }
```

The code meets the specification.

Formal Verification Request 55

ERC20Detailed symbol correctness

09, Jul 2019
1.28 ms

Line 516-518 in File ez365.sol

```
516     /*@CTK "ERC20Detailed symbol correctness"
517       @post __return == _symbol
518      */
```

Line 519-521 in File ez365.sol

```
519     function symbol() public pure returns (string memory) {
520         return _symbol;
521     }
```

The code meets the specification.

Formal Verification Request 56

If method completes, integer overflow would not happen.

09, Jul 2019
17.56 ms

Line 526 in File ez365.sol

```
526     //@CTK NO_OVERFLOW
```

Line 532-534 in File ez365.sol

```
532     function decimals() public pure returns (uint256) {
533         return _decimals;
534     }
```

The code meets the specification.

Formal Verification Request 57

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

09, Jul 2019
0.77 ms

Line 527 in File ez365.sol

```
527     //@CTK NO_BUF_OVERFLOW
```

Line 532-534 in File ez365.sol

```
532     function decimals() public pure returns (uint256) {
533         return _decimals;
534     }
```

The code meets the specification.

Formal Verification Request 58

Method will not encounter an assertion failure.

09, Jul 2019
1.17 ms

Line 528 in File ez365.sol

```
528     //@CTK NO_ASF
```

Line 532-534 in File ez365.sol

```
532     function decimals() public pure returns (uint256) {
533         return _decimals;
534     }
```

The code meets the specification.

Formal Verification Request 59

ERC20Detailed decimals correctness

09, Jul 2019
0.84 ms

Line 529-531 in File ez365.sol

```
529     /*@CTK "ERC20Detailed decimals correctness"
530       @post __return == _decimals
531      */
```

Line 532-534 in File ez365.sol

```
532     function decimals() public pure returns (uint256) {
533         return _decimals;
534     }
```

The code meets the specification.

Formal Verification Request 60

If method completes, integer overflow would not happen.

09, Jul 2019
457.56 ms

Line 548 in File ez365.sol

```
548     //@CTK NO_OVERFLOW
```

Line 558-560 in File ez365.sol

```
558     function burn(uint256 value) public {
559         _burn(msg.sender, value);
560     }
```

The code meets the specification.

Formal Verification Request 61

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

09, Jul 2019
14.68 ms

Line 549 in File ez365.sol

```
549     //@CTK NO_BUF_OVERFLOW
```

Line 558-560 in File ez365.sol

```
558     function burn(uint256 value) public {
559         _burn(msg.sender, value);
560     }
```

The code meets the specification.

Formal Verification Request 62

Method will not encounter an assertion failure.

09, Jul 2019
15.81 ms

Line 550 in File ez365.sol

```
550     //@CTK NO_ASF
```

Line 558-560 in File ez365.sol

```
558     function burn(uint256 value) public {
559         _burn(msg.sender, value);
560     }
```

The code meets the specification.

Formal Verification Request 63

burn correctness

09, Jul 2019
189.43 ms

Line 551-557 in File ez365.sol

```
551     /*@CTK "burn correctness"
552       @tag assume_completion
553       @post msg.sender != 0x0
554       @post value <= _balances[msg.sender]
555       @post __post._balances[msg.sender] == _balances[msg.sender] - value
556       @post __post._totalSupply == _totalSupply - value
557      */
```

Line 558-560 in File ez365.sol

```
558     function burn(uint256 value) public {
559         _burn(msg.sender, value);
560     }
```

The code meets the specification.

Formal Verification Request 64

If method completes, integer overflow would not happen.

09, Jul 2019
931.92 ms

Line 567 in File ez365.sol

```
567     //@CTK NO_OVERFLOW
```

Line 578-580 in File ez365.sol

```
578     function burnFrom(address from, uint256 value) public {
579         _burnFrom(from, value);
580     }
```

The code meets the specification.

Formal Verification Request 65

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

09, Jul 2019
28.57 ms

Line 568 in File ez365.sol

```
568     //@CTK NO_BUF_OVERFLOW
```

Line 578-580 in File ez365.sol

```
578     function burnFrom(address from, uint256 value) public {
579         _burnFrom(from, value);
580     }
```

The code meets the specification.

Formal Verification Request 66

Method will not encounter an assertion failure.

09, Jul 2019
23.83 ms

Line 569 in File ez365.sol

```
569     //@CTK NO_ASF
```

Line 578-580 in File ez365.sol

```
578     function burnFrom(address from, uint256 value) public {
579         _burnFrom(from, value);
580     }
```

The code meets the specification.

Formal Verification Request 67

burnFrom correctness

09, Jul 2019
375.8 ms

Line 570-577 in File ez365.sol

```
570     /*@CTK "burnFrom correctness"
571       @tag assume_completion
572       @post from != 0x0
573       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
574       @post __post._balances[from] == _balances[from] - value
575       @post __post._totalSupply == _totalSupply - value
576       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
577      */
```

Line 578-580 in File ez365.sol

```
578     function burnFrom(address from, uint256 value) public {
579         _burnFrom(from, value);
580     }
```

The code meets the specification.

Formal Verification Request 68

If method completes, integer overflow would not happen.

09, Jul 2019
96.87 ms

Line 582 in File ez365.sol

```
582     //@CTK NO_OVERFLOW
```

Line 589-591 in File ez365.sol

```
589     function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
590         _releaseTime = tokenTime;
591     }
```

The code meets the specification.

Formal Verification Request 69

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

09, Jul 2019
1.47 ms

Line 583 in File ez365.sol

```
583     //@CTK NO_BUF_OVERFLOW
```

Line 589-591 in File ez365.sol

```
589     function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
590         _releaseTime = tokenTime;
591     }
```

The code meets the specification.

Formal Verification Request 70

Method will not encounter an assertion failure.

09, Jul 2019
1.67 ms

Line 584 in File ez365.sol

```
584     //@CTK NO_ASF
```

Line 589-591 in File ez365.sol

```
589     function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
590         _releaseTime = tokenTime;
591     }
```

The code meets the specification.

Formal Verification Request 71

updateReleaseTokenTime correctness

09, Jul 2019
5.42 ms

Line 585-588 in File ez365.sol

```
585     /*@CTK "updateReleaseTokenTime correctness"
586       @post _owner != msg.sender -> __reverted
587       @post _owner == msg.sender -> __post._releaseTime == tokenTime
588      */
```

Line 589-591 in File ez365.sol

```
589     function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
590         _releaseTime = tokenTime;
591     }
```

The code meets the specification.

Formal Verification Request 72

If method completes, integer overflow would not happen.

09, Jul 2019
1208.96 ms

Line 602 in File ez365.sol

```
602     //@CTK NO_OVERFLOW
```

Line 615-617 in File ez365.sol

```
615     function transfer(address _to, uint256 _value) public isTokenReleased returns (
            bool) {
616         super.transfer(_to,_value);
617     }
```

The code meets the specification.

Formal Verification Request 73

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

09, Jul 2019
79.55 ms

Line 603 in File ez365.sol

```
603     //@CTK NO_BUF_OVERFLOW
```

Line 615-617 in File ez365.sol

```
615     function transfer(address _to, uint256 _value) public isTokenReleased returns (
            bool) {
616         super.transfer(_to,_value);
617     }
```

The code meets the specification.

Formal Verification Request 74

Method will not encounter an assertion failure.

09, Jul 2019
82.03 ms

Line 604 in File ez365.sol

```
604     //@CTK NO_ASF
```

Line 615-617 in File ez365.sol

```
615     function transfer(address _to, uint256 _value) public isTokenReleased returns (
            bool) {
616         super.transfer(_to,_value);
617     }
```

The code meets the specification.

Formal Verification Request 75

transfer correctness

09, Jul 2019
3580.02 ms

Line 605-614 in File ez365.sol

```
605     /*@CTK FAIL "transfer correctness"
606       @tag assume_completion
607       @post now >= _releaseTime || _owner == msg.sender
608       @post _to != 0x0
609       @post _value <= _balances[msg.sender]
610       @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - _value
611       @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
612       @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
613       @post __return == true
614      */
```

Line 615-617 in File ez365.sol

```
615     function transfer(address _to, uint256 _value) public isTokenReleased returns (
            bool) {
616         super.transfer(_to,_value);
617     }
```

This code violates the specification.

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

```
        "contract_name": "EZ365Token",
        "balance": 0,
        "contract": {
          "_releaseTime": 0,
          "_name": "",
          "_symbol": "",
          "_decimals": 0,
          "_balances": [
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 160,
              "value": 8
            },
            {
              "key": 0,
              "value": 64
            },
            {
              "key": 33,
              "value": 0
            },
            {
              "key": 1,
              "value": 96
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 4,
              "value": 2
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 9,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 32
            }
          ],
          "_allowed": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 2
                },
                {
```

```
                  "key": 2,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 2,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 32
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      _to = 1
      _value = 32
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = false
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
```

```
        "value": 0
      }
    }
    Other = {
      __return = false
      block = {
        "number": 0,
        "timestamp": 0
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "EZ365Token",
          "balance": 0,
          "contract": {
            "_releaseTime": 0,
            "_name": "",
            "_symbol": "",
            "_decimals": 0,
            "_balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 160,
                "value": 8
              },
              {
                "key": 33,
                "value": 0
              },
              {
                "key": 1,
                "value": 128
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": 4,
                "value": 2
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 9,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 32
              }
```

```
          ],
          "_allowed": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 2
                },
                {
                  "key": 2,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 2,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 32
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 76

If method completes, integer overflow would not happen.

09, Jul 2019
2515.28 ms

Line 619 in File ez365.sol

```
619     //@CTK NO_OVERFLOW
```

Line 633-635 in File ez365.sol

```
633   function transferFrom(address _from, address _to, uint256 _value) public
          isTokenReleased returns (bool) {
634     super.transferFrom(_from, _to, _value);
635   }
```

The code meets the specification.

Formal Verification Request 77

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

09, Jul 2019
73.17 ms

Line 620 in File ez365.sol

```
620     //@CTK NO_BUF_OVERFLOW
```

Line 633-635 in File ez365.sol

```
633   function transferFrom(address _from, address _to, uint256 _value) public
          isTokenReleased returns (bool) {
634     super.transferFrom(_from, _to, _value);
635   }
```

The code meets the specification.

Formal Verification Request 78

Method will not encounter an assertion failure.

09, Jul 2019
80.15 ms

Line 621 in File ez365.sol

```
621     //@CTK NO_ASF
```

Line 633-635 in File ez365.sol

```
633   function transferFrom(address _from, address _to, uint256 _value) public
          isTokenReleased returns (bool) {
634     super.transferFrom(_from, _to, _value);
635   }
```

The code meets the specification.

Formal Verification Request 79

transferFrom correctness

09, Jul 2019
63811.34 ms

Line 622-632 in File ez365.sol

```
622     /*@CTK FAIL "transferFrom correctness"
623       @tag assume_completion
624       @post now >= _releaseTime || _owner == msg.sender
625       @post _to != 0x0
626       @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
627       @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
628       @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
629       @post _to == _from -> __post._balances[_from] == _balances[_from]
630       @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
631       @post __return == true
632      */
```

Line 633-635 in File ez365.sol

```
633   function transferFrom(address _from, address _to, uint256 _value) public
          isTokenReleased returns (bool) {
634       super.transferFrom(_from, _to, _value);
635     }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _from = 16
5          _to = 16
6          _value = 2
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": 132,
18           "value": 0
19         }
20     }
21     Other = {
22         __return = false
23         block = {
24           "number": 0,
25           "timestamp": 128
26         }
27     }
28     Address_Map = [
29       {
30         "key": 0,
```

```
      "value": {
        "contract_name": "EZ365Token",
        "balance": 0,
        "contract": {
          "_releaseTime": 16,
          "_name": "",
          "_symbol": "",
          "_decimals": 0,
          "_balances": [
            {
              "key": 0,
              "value": 3
            },
            {
              "key": 5,
              "value": 0
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 8,
              "value": 64
            },
            {
              "key": 16,
              "value": 2
            },
            {
              "key": 24,
              "value": 0
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 132
            }
          ],
          "_allowed": [
            {
              "key": 128,
              "value": [
                {
                  "key": 0,
                  "value": 8
                },
```

```
                {
                  "key": "ALL_OTHERS",
                  "value": 64
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 1,
                  "value": 0
                },
                {
                  "key": 32,
                  "value": 2
                },
                {
                  "key": 0,
                  "value": 4
                },
                {
                  "key": 2,
                  "value": 32
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 129
                }
              ]
            },
            {
              "key": 16,
              "value": [
                {
                  "key": 128,
                  "value": 30
                },
                {
                  "key": 132,
                  "value": 2
                },
                {
                  "key": 0,
                  "value": 8
                },
                {
                  "key": 32,
                  "value": 0
                },
                {
                  "key": 1,
                  "value": 0
                },
                {
                  "key": 144,
                  "value": 0
                },
```

```
                {
                  "key": "ALL_OTHERS",
                  "value": 132
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 255
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
      _from = 16
      _to = 16
      _value = 2
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = false
      __has_returned = false
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 132,
        "value": 0
      }
  }
  Other = {
      __return = false
      block = {
        "number": 0,
        "timestamp": 128
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "EZ365Token",
```

```
        "balance": 0,
        "contract": {
          "_releaseTime": 16,
          "_name": "",
          "_symbol": "",
          "_decimals": 0,
          "_balances": [
            {
              "key": 0,
              "value": 3
            },
            {
              "key": 5,
              "value": 0
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 8,
              "value": 64
            },
            {
              "key": 16,
              "value": 2
            },
            {
              "key": 24,
              "value": 0
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 132
            }
          ],
          "_allowed": [
            {
              "key": 128,
              "value": [
                {
                  "key": 0,
                  "value": 8
                },
                {
                  "key": "ALL_OTHERS",
```

```
                    "value": 64
                  }
                ]
              },
              {
                "key": 0,
                "value": [
                  {
                    "key": 1,
                    "value": 0
                  },
                  {
                    "key": 32,
                    "value": 2
                  },
                  {
                    "key": 0,
                    "value": 4
                  },
                  {
                    "key": 2,
                    "value": 32
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 129
                  }
                ]
              },
              {
                "key": 16,
                "value": [
                  {
                    "key": 128,
                    "value": 30
                  },
                  {
                    "key": 144,
                    "value": 0
                  },
                  {
                    "key": 0,
                    "value": 8
                  },
                  {
                    "key": 32,
                    "value": 0
                  },
                  {
                    "key": 1,
                    "value": 0
                  },
                  {
                    "key": 132,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
```

```
                  "value": 132
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 255
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 80

If method completes, integer overflow would not happen.

09, Jul 2019
892.86 ms

Line 637 in File ez365.sol

```
637     //@CTK NO_OVERFLOW
```

Line 647-649 in File ez365.sol

```
647   function increaseAllowance(address _spender, uint _addedValue) public
          isTokenReleased returns (bool) {
648     super.increaseAllowance(_spender, _addedValue);
649   }
```

The code meets the specification.

Formal Verification Request 81

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

09, Jul 2019
26.45 ms

Line 638 in File ez365.sol

```
638     //@CTK NO_BUF_OVERFLOW
```

Line 647-649 in File ez365.sol

```
647   function increaseAllowance(address _spender, uint _addedValue) public
          isTokenReleased returns (bool) {
648     super.increaseAllowance(_spender, _addedValue);
649   }
```

The code meets the specification.

Formal Verification Request 82

Method will not encounter an assertion failure.

09, Jul 2019
28.92 ms

Line 639 in File ez365.sol

```
639   //@CTK NO_ASF
```

Line 647-649 in File ez365.sol

```
647   function increaseAllowance(address _spender, uint _addedValue) public
          isTokenReleased returns (bool) {
648     super.increaseAllowance(_spender, _addedValue);
649   }
```

The code meets the specification.

Formal Verification Request 83

increaseAllowance correctness

09, Jul 2019
1203.58 ms

Line 640-646 in File ez365.sol

```
640   /*@CTK FAIL "increaseAllowance correctness"
641     @tag assume_completion
642     @post now >= _releaseTime || _owner == msg.sender
643     @post _spender != 0x0
644     @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
            _addedValue
645     @post __return == true
646   */
```

Line 647-649 in File ez365.sol

```
647   function increaseAllowance(address _spender, uint _addedValue) public
          isTokenReleased returns (bool) {
648     super.increaseAllowance(_spender, _addedValue);
649   }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _addedValue = 0
```

```
        _spender = 16
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 2,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "EZ365Token",
          "balance": 0,
          "contract": {
            "_releaseTime": 0,
            "_name": "",
            "_symbol": "",
            "_decimals": 0,
            "_balances": [
              {
                "key": 1,
                "value": 128
              },
              {
                "key": 0,
                "value": 64
              },
              {
                "key": 4,
                "value": 0
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 32,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
```

```
          ],
          "_allowed": [
            {
              "key": 1,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": 8,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": 2,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": 16,
                  "value": 64
                },
                {
                  "key": 2,
                  "value": 4
                },
                {
                  "key": 34,
                  "value": 0
                },
                {
                  "key": 32,
                  "value": 2
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
```

```
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 2
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
    Input = {
        _addedValue = 0
        _spender = 16
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 2,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "EZ365Token",
          "balance": 0,
          "contract": {
            "_releaseTime": 0,
            "_name": "",
```

```
          "_symbol": "",
          "_decimals": 0,
          "_balances": [
            {
              "key": 1,
              "value": 128
            },
            {
              "key": 0,
              "value": 64
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 16
            }
          ],
          "_allowed": [
            {
              "key": 1,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": 8,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
```

```
            },
            {
              "key": 2,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": 16,
                  "value": 64
                },
                {
                  "key": 2,
                  "value": 4
                },
                {
                  "key": 34,
                  "value": 0
                },
                {
                  "key": 32,
                  "value": 2
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 2
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```


Formal Verification Request 84

If method completes, integer overflow would not happen.

09, Jul 2019
462.42 ms

Line 651 in File ez365.sol

```
651     //@CTK NO_OVERFLOW
```

Line 661-663 in File ez365.sol

```
661   function decreaseAllowance(address _spender, uint _subtractedValue) public
          isTokenReleased returns (bool) {
662     super.decreaseAllowance(_spender, _subtractedValue);
663     }
```

The code meets the specification.

Formal Verification Request 85

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

09, Jul 2019
25.27 ms

Line 652 in File ez365.sol

```
652     //@CTK NO_BUF_OVERFLOW
```

Line 661-663 in File ez365.sol

```
661   function decreaseAllowance(address _spender, uint _subtractedValue) public
          isTokenReleased returns (bool) {
662     super.decreaseAllowance(_spender, _subtractedValue);
663     }
```

The code meets the specification.

Formal Verification Request 86

Method will not encounter an assertion failure.

09, Jul 2019
28.43 ms

Line 653 in File ez365.sol

```
653     //@CTK NO_ASF
```

Line 661-663 in File ez365.sol

```
661   function decreaseAllowance(address _spender, uint _subtractedValue) public
          isTokenReleased returns (bool) {
662     super.decreaseAllowance(_spender, _subtractedValue);
663     }
```

The code meets the specification.

Formal Verification Request 87

decreaseAllowance correctness

09, Jul 2019
3552.52 ms

Line 654-660 in File ez365.sol

```
654     /*@CTK FAIL "decreaseAllowance correctness"
655       @tag assume_completion
656       @post now >= _releaseTime || _owner == msg.sender
657       @post _spender != 0x0
658       @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] -
              _subtractedValue
659       @post __return == true
660      */
```

Line 661-663 in File ez365.sol

```
661     function decreaseAllowance(address _spender, uint _subtractedValue) public
            isTokenReleased returns (bool) {
662       super.decreaseAllowance(_spender, _subtractedValue);
663     }
```

This code violates the specification.

```
1   Counter Example:
2   Before Execution:
3       Input = {
4           _spender = 128
5           _subtractedValue = 0
6       }
7       This = 0
8       Internal = {
9           __has_assertion_failure = false
10          __has_buf_overflow = false
11          __has_overflow = false
12          __has_returned = false
13          __reverted = false
14          msg = {
15            "gas": 0,
16            "sender": 32,
17            "value": 0
18          }
19      }
20      Other = {
21          __return = false
22          block = {
23            "number": 0,
24            "timestamp": 0
25          }
26      }
27      Address_Map = [
28        {
29          "key": 0,
30          "value": {
31            "contract_name": "EZ365Token",
32            "balance": 0,
33            "contract": {
```

```
          "_releaseTime": 0,
          "_name": "",
          "_symbol": "",
          "_decimals": 0,
          "_balances": [
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 1,
              "value": 2
            },
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 16,
              "value": 2
            },
            {
              "key": 4,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "_allowed": [
            {
              "key": 8,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": 16,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
```

```
                }
              ]
            },
            {
              "key": 32,
              "value": [
                {
                  "key": 2,
                  "value": 0
                },
                {
                  "key": 128,
                  "value": 0
                },
                {
                  "key": 0,
                  "value": 1
                },
                {
                  "key": 32,
                  "value": 64
                },
                {
                  "key": 33,
                  "value": 0
                },
                {
                  "key": 8,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 32
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
```

```
    Input = {
        _spender = 128
        _subtractedValue = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 32,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "EZ365Token",
          "balance": 0,
          "contract": {
            "_releaseTime": 0,
            "_name": "",
            "_symbol": "",
            "_decimals": 0,
            "_balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 1,
                "value": 2
              },
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 16,
                "value": 2
              },
              {
                "key": 4,
                "value": 32
              },
              {
                "key": "ALL_OTHERS",
```

```
              "value": 128
            }
          ],
          "_allowed": [
            {
              "key": 8,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 16
                }
              ]
            },
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": 16,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": 32,
              "value": [
                {
                  "key": 2,
                  "value": 0
                },
                {
                  "key": 128,
                  "value": 0
                },
                {
                  "key": 0,
                  "value": 1
                },
                {
                  "key": 32,
                  "value": 64
                },
                {
                  "key": 33,
                  "value": 0
                },
                {
```

```
                  "key": 8,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            }
          ],
          "_totalSupply": 0,
          "_owner": 32
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Source Code with CertiK Labels

File ez365.sol

```
1  /**
2   *Submitted for verification at Etherscan.io on 2019-04-12
3  */
4
5  pragma solidity ^0.5.2;
6  /**
7   * @title Ownable
8   * @dev The Ownable contract has an owner address, and provides basic authorization
       control
9   * functions, this simplifies the implementation of "user permissions".
10  */
11 contract Ownable {
12     address private _owner;
13
14     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
           );
15
16     /**
17      * @dev The Ownable constructor sets the original `owner` of the contract to the
            sender
18      * account.
19      */
20     /*@CTK Ownable
21       @post __post._owner == msg.sender
22      */
23     constructor () internal {
24         _owner = msg.sender;
25         emit OwnershipTransferred(address(0), _owner);
26     }
27
28     /**
29      * @return the address of the owner.
30      */
31     /*@CTK owner
32       @post __return == _owner
33      */
34     function owner() public view returns (address) {
35         return _owner;
36     }
37
38     /**
39      * @dev Throws if called by any account other than the owner.
40      */
41     modifier onlyOwner() {
42         require(isOwner());
43         _;
44     }
45
46     /**
47      * @return true if `msg.sender` is the owner of the contract.
48      */
49     /*@CTK isOwner
50       @post __return == (msg.sender == _owner)
51      */
```

```
52      function isOwner() public view returns (bool) {
53          return msg.sender == _owner;
54      }
55
56      /**
57       * @dev Allows the current owner to relinquish control of the contract.
58       * It will not be possible to call the functions with the ‘onlyOwner‘
59       * modifier anymore.
60       * @notice Renouncing ownership will leave the contract without an owner,
61       * thereby removing any functionality that is only available to the owner.
62       */
63      /*@CTK renounceOwnership
64        @tag assume_completion
65        @post _owner == msg.sender
66        @post __post._owner == address(0)
67       */
68      function renounceOwnership() public onlyOwner {
69          emit OwnershipTransferred(_owner, address(0));
70          _owner = address(0);
71      }
72
73      /**
74       * @dev Allows the current owner to transfer control of the contract to a newOwner
           .
75       * @param newOwner The address to transfer ownership to.
76       */
77      /*@CTK transferOwnership
78        @tag assume_completion
79        @post _owner == msg.sender
80       */
81      function transferOwnership(address newOwner) public onlyOwner {
82          _transferOwnership(newOwner);
83      }
84
85      /**
86       * @dev Transfers control of the contract to a newOwner.
87       * @param newOwner The address to transfer ownership to.
88       */
89      /*@CTK _transferOwnership
90        @tag assume_completion
91        @post newOwner != address(0)
92        @post __post._owner == newOwner
93       */
94      function _transferOwnership(address newOwner) internal {
95          require(newOwner != address(0));
96          emit OwnershipTransferred(_owner, newOwner);
97          _owner = newOwner;
98      }
99  }
100 /**
101  * @title SafeMath
102  * @dev Unsigned math operations with safety checks that revert on error.
103  */
104 library SafeMath {
105     /**
106      * @dev Multiplies two unsigned integers, reverts on overflow.
107      */
108     /*@CTK "SafeMath mul"
```

```
109       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
110       @post !__reverted -> __return == a * b
111       @post !__reverted == !__has_overflow
112       @post !(__has_buf_overflow)
113       @post !(__has_assertion_failure)
114     */
115     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
116         // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
117         // benefit is lost if 'b' is also tested.
118         // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
119         if (a == 0) {
120             return 0;
121         }
122
123         uint256 c = a * b;
124         require(c / a == b);
125
126         return c;
127     }
128
129     /**
130      * @dev Integer division of two unsigned integers truncating the quotient, reverts
            on division by zero.
131      */
132     /*@CTK "SafeMath div"
133       @post b != 0 -> !__reverted
134       @post !__reverted -> __return == a / b
135       @post !__reverted -> !__has_overflow
136       @post !(__has_buf_overflow)
137       @post !(__has_assertion_failure)
138     */
139     function div(uint256 a, uint256 b) internal pure returns (uint256) {
140         // Solidity only automatically asserts when dividing by 0
141         require(b > 0);
142         uint256 c = a / b;
143         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
144
145         return c;
146     }
147
148     /**
149      * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend
            is greater than minuend).
150      */
151     /*@CTK "SafeMath sub"
152       @post (a < b) == __reverted
153       @post !__reverted -> __return == a - b
154       @post !__reverted -> !__has_overflow
155       @post !(__has_buf_overflow)
156       @post !(__has_assertion_failure)
157     */
158     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
159         require(b <= a);
160         uint256 c = a - b;
161
162         return c;
163     }
164
```

```
165     /**
166      * @dev Adds two unsigned integers, reverts on overflow.
167      */
168     /*@CTK "SafeMath add"
169       @post (a + b < a || a + b < b) == __reverted
170       @post !__reverted -> __return == a + b
171       @post !__reverted -> !__has_overflow
172       @post !(__has_buf_overflow)
173       @post !(__has_assertion_failure)
174      */
175     function add(uint256 a, uint256 b) internal pure returns (uint256) {
176         uint256 c = a + b;
177         require(c >= a);
178 
179         return c;
180     }
181 
182     /**
183      * @dev Divides two unsigned integers and returns the remainder (unsigned integer
             modulo),
184      * reverts when dividing by zero.
185      */
186     /*@CTK "SafeMath mod"
187       @post b != 0 -> !__reverted
188       @post !__reverted -> __return == a % b
189       @post !__reverted -> !__has_overflow
190       @post !(__has_buf_overflow)
191       @post !(__has_assertion_failure)
192      */
193     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
194         require(b != 0);
195         return a % b;
196     }
197 }
198 /**
199  * @title ERC20 interface
200  * @dev see https://eips.ethereum.org/EIPS/eip-20
201  */
202 interface IERC20 {
203     function transfer(address to, uint256 value) external returns (bool);
204 
205     function approve(address spender, uint256 value) external returns (bool);
206 
207     function transferFrom(address from, address to, uint256 value) external returns (
            bool);
208 
209     function totalSupply() external view returns (uint256);
210 
211     function balanceOf(address who) external view returns (uint256);
212 
213     function allowance(address owner, address spender) external view returns (uint256)
            ;
214 
215     event Transfer(address indexed from, address indexed to, uint256 value);
216 
217     event Approval(address indexed owner, address indexed spender, uint256 value);
218 }
219 
```

```
220 /**
221  * @title Standard ERC20 token
222  *
223  * @dev Implementation of the basic standard token.
224  * https://eips.ethereum.org/EIPS/eip-20
225  * Originally based on code by FirstBlood:
226  * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.
         sol
227  *
228  * This implementation emits additional Approval events, allowing applications to
         reconstruct the allowance status for
229  * all accounts just by listening to said events. Note that this isn't required by the
         specification, and other
230  * compliant implementations may not do it.
231  */
232 contract ERC20 is IERC20, Ownable {
233     using SafeMath for uint256;
234 
235     mapping (address => uint256) private _balances;
236 
237     mapping (address => mapping (address => uint256)) private _allowed;
238 
239     uint256 private _totalSupply;
240 
241     /**
242      * @dev Total number of tokens in existence.
243      */
244     //@CTK NO_OVERFLOW
245     //@CTK NO_BUF_OVERFLOW
246     //@CTK NO_ASF
247     /*@CTK "totalSupply correctness"
248       @post __return == _totalSupply
249      */
250     function totalSupply() public view returns (uint256) {
251         return _totalSupply;
252     }
253 
254     /**
255      * @dev Gets the balance of the specified address.
256      * @param owner The address to query the balance of.
257      * @return A uint256 representing the amount owned by the passed address.
258      */
259     //@CTK NO_OVERFLOW
260     //@CTK NO_BUF_OVERFLOW
261     //@CTK NO_ASF
262     /*@CTK "balanceOf correctness"
263       @post __return == _balances[owner]
264      */
265     function balanceOf(address owner) public view returns (uint256) {
266         return _balances[owner];
267     }
268 
269     /**
270      * @dev Function to check the amount of tokens that an owner allowed to a spender.
271      * @param owner address The address which owns the funds.
272      * @param spender address The address which will spend the funds.
273      * @return A uint256 specifying the amount of tokens still available for the
           spender.
```

```
274      */
275     //@CTK NO_OVERFLOW
276     //@CTK NO_BUF_OVERFLOW
277     //@CTK NO_ASF
278     /*@CTK "allowance correctness"
279       @post __return == _allowed[owner][spender]
280      */
281     function allowance(address owner, address spender) public view returns (uint256) {
282         return _allowed[owner][spender];
283     }
284
285     /**
286      * @dev Transfer token to a specified address.
287      * @param to The address to transfer to.
288      * @param value The amount to be transferred.
289      */
290     //@CTK NO_OVERFLOW
291     //@CTK NO_BUF_OVERFLOW
292     //@CTK NO_ASF
293     /*@CTK "transfer correctness"
294       @tag assume_completion
295       @post to != 0x0
296       @post value <= _balances[msg.sender]
297       @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - value
298       @post to != msg.sender -> __post._balances[to] == _balances[to] + value
299       @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
300       @post __return == true
301      */
302     function transfer(address to, uint256 value) public returns (bool) {
303         _transfer(msg.sender, to, value);
304         return true;
305     }
306
307     /**
308      * @dev Approve the passed address to spend the specified amount of tokens on
             behalf of msg.sender.
309      * Beware that changing an allowance with this method brings the risk that someone
              may use both the old
310      * and the new allowance by unfortunate transaction ordering. One possible
             solution to mitigate this
311      * race condition is to first reduce the spender's allowance to 0 and set the
             desired value afterwards:
312      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
313      * @param spender The address which will spend the funds.
314      * @param value The amount of tokens to be spent.
315      */
316     //@CTK NO_OVERFLOW
317     //@CTK NO_BUF_OVERFLOW
318     //@CTK NO_ASF
319     /*@CTK "approve correctness"
320       @pre msg.sender != 0x0
321       @post spender == 0x0 -> __reverted
322       @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
323      */
324     function approve(address spender, uint256 value) public returns (bool) {
325         _approve(msg.sender, spender, value);
326         return true;
```

```
327     }
328
329     /**
330      * @dev Transfer tokens from one address to another.
331      * Note that while this function emits an Approval event, this is not required as
             per the specification,
332      * and other compliant implementations may not emit the event.
333      * @param from address The address which you want to send tokens from
334      * @param to address The address which you want to transfer to
335      * @param value uint256 the amount of tokens to be transferred
336      */
337     //@CTK NO_OVERFLOW
338     //@CTK NO_BUF_OVERFLOW
339     //@CTK NO_ASF
340     /*@CTK "transferFrom correctness"
341       @tag assume_completion
342       @post to != 0x0
343       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
344       @post to != from -> __post._balances[from] == _balances[from] - value
345       @post to != from -> __post._balances[to] == _balances[to] + value
346       @post to == from -> __post._balances[from] == _balances[from]
347       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
348       @post __return == true
349      */
350     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
351         _transfer(from, to, value);
352         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
353         return true;
354     }
355
356     /**
357      * @dev Increase the amount of tokens that an owner allowed to a spender.
358      * approve should be called when _allowed[msg.sender][spender] == 0. To increment
359      * allowed value is better to use this function to avoid 2 calls (and wait until
360      * the first transaction is mined)
361      * From MonolithDAO Token.sol
362      * Emits an Approval event.
363      * @param spender The address which will spend the funds.
364      * @param addedValue The amount of tokens to increase the allowance by.
365      */
366     //@CTK NO_OVERFLOW
367     //@CTK NO_BUF_OVERFLOW
368     //@CTK NO_ASF
369     /*@CTK "increaseAllowance correctness"
370       @tag assume_completion
371       @post spender != 0x0
372       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
             addedValue
373       @post __return == true
374     */
375     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
376         _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
377         return true;
378     }
379
380     /**
```

```
381      * @dev Decrease the amount of tokens that an owner allowed to a spender.
382      * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
383      * allowed value is better to use this function to avoid 2 calls (and wait until
384      * the first transaction is mined)
385      * From MonolithDAO Token.sol
386      * Emits an Approval event.
387      * @param spender The address which will spend the funds.
388      * @param subtractedValue The amount of tokens to decrease the allowance by.
389      */
390     //@CTK NO_OVERFLOW
391     //@CTK NO_BUF_OVERFLOW
392     //@CTK NO_ASF
393     /*@CTK "decreaseAllowance correctness"
394       @tag assume_completion
395       @post spender != 0x0
396       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
              subtractedValue
397       @post __return == true
398      */
399     function decreaseAllowance(address spender, uint256 subtractedValue) public
          returns (bool) {
400         _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
              ));
401         return true;
402     }
403
404     /**
405      * @dev Transfer token for a specified addresses.
406      * @param from The address to transfer from.
407      * @param to The address to transfer to.
408      * @param value The amount to be transferred.
409      */
410     function _transfer(address from, address to, uint256 value) internal {
411         require(to != address(0));
412
413         _balances[from] = _balances[from].sub(value);
414         _balances[to] = _balances[to].add(value);
415         emit Transfer(from, to, value);
416     }
417
418     /**
419      * @dev Internal function that mints an amount of the token and assigns it to
420      * an account. This encapsulates the modification of balances such that the
421      * proper events are emitted.
422      * @param account The account that will receive the created tokens.
423      * @param value The amount that will be created.
424      */
425     //@CTK NO_OVERFLOW
426     //@CTK NO_BUF_OVERFLOW
427     //@CTK NO_ASF
428     /*@CTK "_mint correctness"
429       @tag assume_completion
430       @post account != 0x0
431       @post __post._balances[account] == _balances[account] + value
432       @post __post._totalSupply == _totalSupply + value
433      */
434     function _mint(address account, uint256 value) internal {
435         require(account != address(0));
```

```
436
437         _totalSupply = _totalSupply.add(value);
438         _balances[account] = _balances[account].add(value);
439         emit Transfer(address(0), account, value);
440     }
441
442     /**
443      * @dev Internal function that burns an amount of the token of a given
444      * account.
445      * @param account The account whose tokens will be burnt.
446      * @param value The amount that will be burnt.
447      */
448     function _burn(address account, uint256 value) internal {
449         require(account != address(0));
450
451         _totalSupply = _totalSupply.sub(value);
452         _balances[account] = _balances[account].sub(value);
453         emit Transfer(account, address(0), value);
454     }
455
456     /**
457      * @dev Approve an address to spend another addresses' tokens.
458      * @param owner The address that owns the tokens.
459      * @param spender The address that will spend the tokens.
460      * @param value The number of tokens that can be spent.
461      */
462     function _approve(address owner, address spender, uint256 value) internal {
463         require(spender != address(0));
464         require(owner != address(0));
465
466         _allowed[owner][spender] = value;
467         emit Approval(owner, spender, value);
468     }
469
470     /**
471      * @dev Internal function that burns an amount of the token of a given
472      * account, deducting from the sender's allowance for said account. Uses the
473      * internal burn function.
474      * Emits an Approval event (reflecting the reduced allowance).
475      * @param account The account whose tokens will be burnt.
476      * @param value The amount that will be burnt.
477      */
478     function _burnFrom(address account, uint256 value) internal {
479         _burn(account, value);
480         _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
481     }
482 }
483
484
485 /**
486  * @title ERC20Detailed token
487  * @dev The decimals are only for visualization purposes.
488  * All the operations are done using the smallest and indivisible token unit,
489  * just as on Ethereum all the operations are done in wei.
490  */
491 contract ERC20Detailed is ERC20 {
492     string constant private _name = "EZ365";
493     string constant private _symbol = "EZ365";
```

```
494     uint256 constant private _decimals = 18;
495
496
497     /**
498      * @return the name of the token.
499      */
500     //@CTK NO_OVERFLOW
501     //@CTK NO_BUF_OVERFLOW
502     //@CTK NO_ASF
503     /*@CTK "ERC20Detailed name correctness"
504       @post __return == _name
505      */
506     function name() public pure returns (string memory) {
507         return _name;
508     }
509
510     /**
511      * @return the symbol of the token.
512      */
513     //@CTK NO_OVERFLOW
514     //@CTK NO_BUF_OVERFLOW
515     //@CTK NO_ASF
516     /*@CTK "ERC20Detailed symbol correctness"
517       @post __return == _symbol
518      */
519     function symbol() public pure returns (string memory) {
520         return _symbol;
521     }
522
523     /**
524      * @return the number of decimals of the token.
525      */
526     //@CTK NO_OVERFLOW
527     //@CTK NO_BUF_OVERFLOW
528     //@CTK NO_ASF
529     /*@CTK "ERC20Detailed decimals correctness"
530       @post __return == _decimals
531      */
532     function decimals() public pure returns (uint256) {
533         return _decimals;
534     }
535 }
536 contract EZ365Token is ERC20Detailed {
537
538     uint256 public _releaseTime;
539     constructor() public {
540         uint256 totalSupply = 200000000 * (10 ** decimals());
541         _mint(msg.sender,totalSupply);
542         _releaseTime = block.timestamp + 365 days;
543     }
544      /**
545      * @dev Burns a specific amount of tokens.
546      * @param value The amount of token to be burned.
547      */
548     //@CTK NO_OVERFLOW
549     //@CTK NO_BUF_OVERFLOW
550     //@CTK NO_ASF
551     /*@CTK "burn correctness"
```

```
552       @tag assume_completion
553       @post msg.sender != 0x0
554       @post value <= _balances[msg.sender]
555       @post __post._balances[msg.sender] == _balances[msg.sender] - value
556       @post __post._totalSupply == _totalSupply - value
557      */
558     function burn(uint256 value) public {
559         _burn(msg.sender, value);
560     }
561
562     /**
563      * @dev Burns a specific amount of tokens from the target address and decrements
             allowance.
564      * @param from address The account whose tokens will be burned.
565      * @param value uint256 The amount of token to be burned.
566      */
567     //@CTK NO_OVERFLOW
568     //@CTK NO_BUF_OVERFLOW
569     //@CTK NO_ASF
570     /*@CTK "burnFrom correctness"
571       @tag assume_completion
572       @post from != 0x0
573       @post value <= _balances[from] && value <= _allowed[from][msg.sender]
574       @post __post._balances[from] == _balances[from] - value
575       @post __post._totalSupply == _totalSupply - value
576       @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
577      */
578     function burnFrom(address from, uint256 value) public {
579         _burnFrom(from, value);
580     }
581
582     //@CTK NO_OVERFLOW
583     //@CTK NO_BUF_OVERFLOW
584     //@CTK NO_ASF
585     /*@CTK "updateReleaseTokenTime correctness"
586       @post _owner != msg.sender -> __reverted
587       @post _owner == msg.sender -> __post._releaseTime == tokenTime
588      */
589     function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
590         _releaseTime = tokenTime;
591     }
592
593     modifier isTokenReleased () {
594         if (isOwner()){
595            _;
596         }else{
597             require(block.timestamp >= _releaseTime);
598             _;
599         }
600     }
601
602     //@CTK NO_OVERFLOW
603     //@CTK NO_BUF_OVERFLOW
604     //@CTK NO_ASF
605     /*@CTK FAIL "transfer correctness"
606       @tag assume_completion
607       @post now >= _releaseTime || _owner == msg.sender
608       @post _to != 0x0
```

```
609       @post _value <= _balances[msg.sender]
610       @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - _value
611       @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
612       @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
613       @post __return == true
614      */
615     function transfer(address _to, uint256 _value) public isTokenReleased returns (
              bool) {
616         super.transfer(_to,_value);
617     }
618
619     //@CTK NO_OVERFLOW
620     //@CTK NO_BUF_OVERFLOW
621     //@CTK NO_ASF
622     /*@CTK FAIL "transferFrom correctness"
623       @tag assume_completion
624       @post now >= _releaseTime || _owner == msg.sender
625       @post _to != 0x0
626       @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
627       @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
628       @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
629       @post _to == _from -> __post._balances[_from] == _balances[_from]
630       @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
631       @post __return == true
632      */
633     function transferFrom(address _from, address _to, uint256 _value) public
              isTokenReleased returns (bool) {
634       super.transferFrom(_from, _to, _value);
635     }
636
637     //@CTK NO_OVERFLOW
638     //@CTK NO_BUF_OVERFLOW
639     //@CTK NO_ASF
640     /*@CTK FAIL "increaseAllowance correctness"
641       @tag assume_completion
642       @post now >= _releaseTime || _owner == msg.sender
643       @post _spender != 0x0
644       @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
              _addedValue
645       @post __return == true
646      */
647     function increaseAllowance(address _spender, uint _addedValue) public
              isTokenReleased returns (bool) {
648       super.increaseAllowance(_spender, _addedValue);
649     }
650
651     //@CTK NO_OVERFLOW
652     //@CTK NO_BUF_OVERFLOW
653     //@CTK NO_ASF
654     /*@CTK FAIL "decreaseAllowance correctness"
655       @tag assume_completion
656       @post now >= _releaseTime || _owner == msg.sender
657       @post _spender != 0x0
658       @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] -
              _subtractedValue
659       @post __return == true
660      */
```

```
661     function decreaseAllowance(address _spender, uint _subtractedValue) public
            isTokenReleased returns (bool) {
662       super.decreaseAllowance(_spender, _subtractedValue);
663     }
664 }
```