CertiK Audit Report For ez365

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

Contents

Disclaimer

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

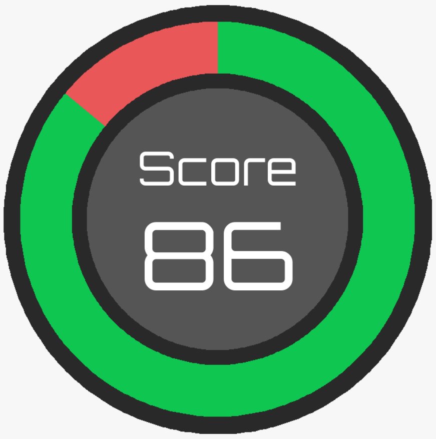
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

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

```
pragma solidity ^0.5.2;
```

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

TIMESTAMP_DEPENDENCY

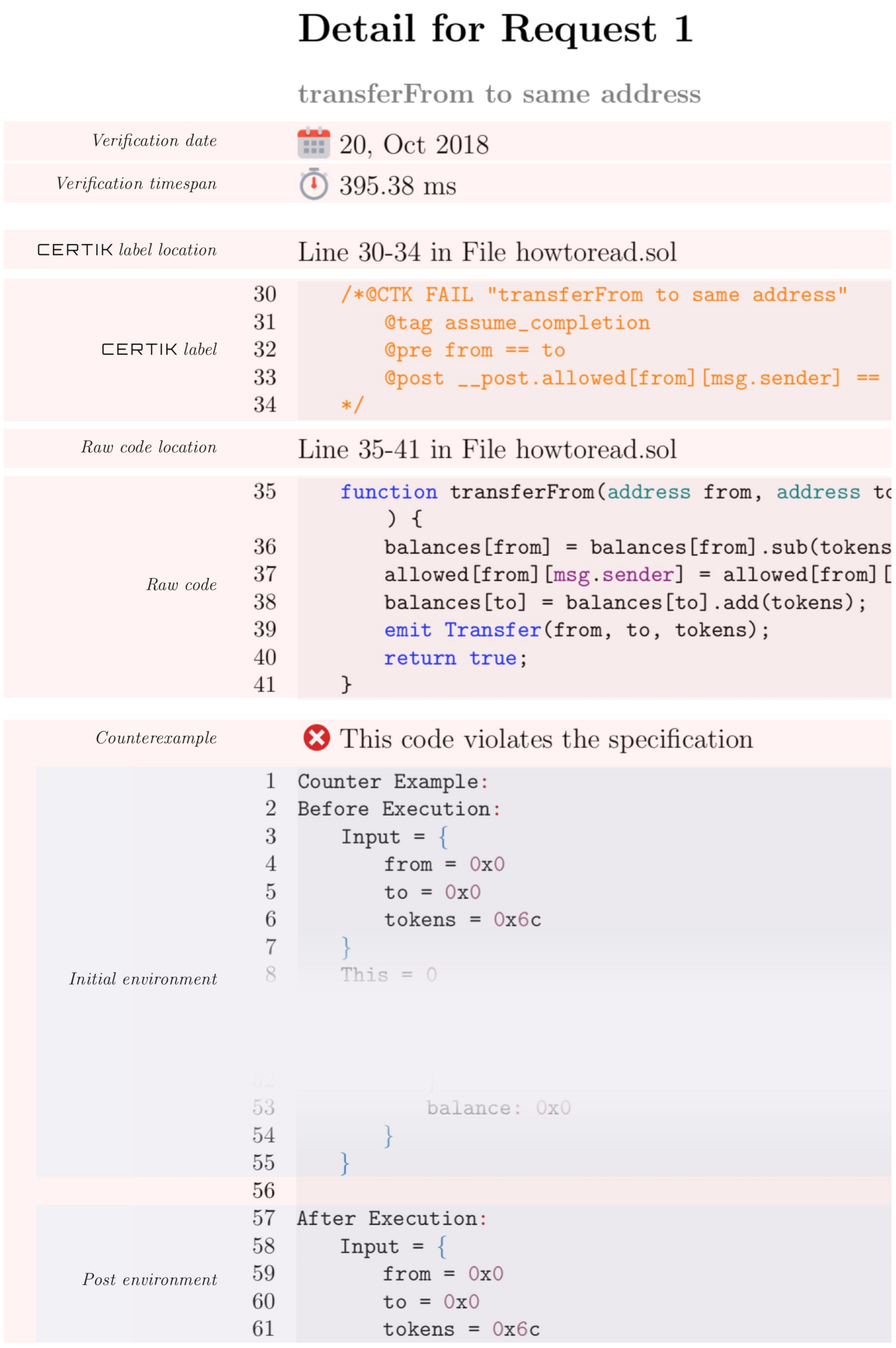
Line 597 in File ez365.sol

```
            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

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

Line 23-26 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 2

owner

📅 09, Jul 2019
⏱ 22.38 ms

Line 31-33 in File ez365.sol

```
    /*@CTK owner
      @post __return == _owner
     */
```

Line 34-36 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 3

isOwner

📅 09, Jul 2019
⏱ 18.56 ms

Line 49-51 in File ez365.sol

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

Line 52-54 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 4

renounceOwnership

09, Jul 2019
104.68 ms

Line 63-67 in File ez365.sol

```
    /*@CTK renounceOwnership
      @tag assume_completion
      @post _owner == msg.sender
      @post __post._owner == address(0)
     */
```

Line 68-71 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 5

transferOwnership

09, Jul 2019
207.33 ms

Line 77-80 in File ez365.sol

```
    /*@CTK transferOwnership
      @tag assume_completion
      @post _owner == msg.sender
     */
```

Line 81-83 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 6

_transferOwnership

📅 09, Jul 2019
⏱ 2.6 ms

Line 89-93 in File ez365.sol

```
    /*@CTK _transferOwnership
      @tag assume_completion
      @post newOwner != address(0)
      @post __post._owner == newOwner
     */
```

Line 94-98 in File ez365.sol

```
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0));
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
```

✓ The code meets the specification.

Formal Verification Request 7

SafeMath mul

📅 09, Jul 2019
⏱ 854.1 ms

Line 108-114 in File ez365.sol

```
    /*@CTK "SafeMath mul"
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
```

Line 115-127 in File ez365.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 8

SafeMath div

📅 09, Jul 2019
⏱ 58.43 ms

Line 132-138 in File ez365.sol

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

Line 139-146 in File ez365.sol

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

        return c;
    }
```

✅ The code meets the specification.

Formal Verification Request 9

SafeMath sub

📅 09, Jul 2019
⏱ 46.74 ms

Line 151-157 in File ez365.sol

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

Line 158-163 in File ez365.sol

```
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }
```

✅ The code meets the specification.

Formal Verification Request 10

SafeMath add

09, Jul 2019
70.34 ms

Line 168-174 in File ez365.sol

```
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> __return == a + b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
```

Line 175-180 in File ez365.sol

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

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 11

SafeMath mod

09, Jul 2019
43.95 ms

Line 186-192 in File ez365.sol

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

Line 193-196 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 250-252 in File ez365.sol

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

✅ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 250-252 in File ez365.sol

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

✅ 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

```
    //@CTK NO_ASF
```

Line 250-252 in File ez365.sol

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

✅ The code meets the specification.

Formal Verification Request 15

totalSupply correctness

09, Jul 2019
0.78 ms

Line 247-249 in File ez365.sol

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

Line 250-252 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 265-267 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 265-267 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 265-267 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 19

balanceOf correctness

📅 09, Jul 2019
⏱ 0.78 ms

Line 262-264 in File ez365.sol

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

Line 265-267 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 281-283 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 281-283 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 281-283 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 23

allowance correctness

09, Jul 2019
1.2 ms

Line 278-280 in File ez365.sol

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

Line 281-283 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 302-305 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 302-305 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 302-305 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 27

transfer correctness

📅 09, Jul 2019
⏱ 163.32 ms

Line 293-301 in File ez365.sol

```
    /*@CTK "transfer correctness"
      @tag assume_completion
      @post to != 0x0
      @post value <= _balances[msg.sender]
      @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
          - value
      @post to != msg.sender -> __post._balances[to] == _balances[to] + value
      @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
      @post __return == true
     */
```

Line 302-305 in File ez365.sol

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

✅ 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

```
    //@CTK NO_OVERFLOW
```

Line 324-327 in File ez365.sol

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

✅ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 324-327 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 324-327 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 31

approve correctness

09, Jul 2019
7.91 ms

Line 319-323 in File ez365.sol

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

Line 324-327 in File ez365.sol

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

✓ 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

```
//@CTK NO_OVERFLOW
```

Line 350-354 in File ez365.sol

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

✓ 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

```
//@CTK NO_BUF_OVERFLOW
```

Line 350-354 in File ez365.sol

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

✓ 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

```
//@CTK NO_ASF
```

Line 350-354 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 35

transferFrom correctness

09, Jul 2019
662.77 ms

Line 340-349 in File ez365.sol

```
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post to != 0x0
  @post value <= _balances[from] && value <= _allowed[from][msg.sender]
  @post to != from -> __post._balances[from] == _balances[from] - value
  @post to != from -> __post._balances[to] == _balances[to] + value
  @post to == from -> __post._balances[from] == _balances[from]
  @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
  @post __return == true
 */
```

Line 350-354 in File ez365.sol

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

✓ 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

```
//@CTK NO_OVERFLOW
```

Line 375-378 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 375-378 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 375-378 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 39

increaseAllowance correctness

09, Jul 2019
7.98 ms

Line 369-374 in File ez365.sol

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

Line 375-378 in File ez365.sol

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

✓ 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

```
//@CTK NO_OVERFLOW
```

Line 399-402 in File ez365.sol

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

✓ 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

```
//@CTK NO_BUF_OVERFLOW
```

Line 399-402 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 399-402 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 43

decreaseAllowance correctness

09, Jul 2019
15.16 ms

Line 393-398 in File ez365.sol

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

Line 399-402 in File ez365.sol

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

✓ 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

```
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }
```

✓ The code meets the specification.

Formal Verification Request 47

_mint correctness

09, Jul 2019
92.66 ms

Line 428-433 in File ez365.sol

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

Line 434-440 in File ez365.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }
```

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 506-508 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 506-508 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 506-508 in File ez365.sol

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

✓ 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

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

Line 506-508 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 519-521 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 519-521 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 519-521 in File ez365.sol

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

✓ 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

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

Line 519-521 in File ez365.sol

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

✅ 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

```
    //@CTK NO_OVERFLOW
```

Line 532-534 in File ez365.sol

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

✅ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 532-534 in File ez365.sol

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

✅ 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

```
    //@CTK NO_ASF
```

Line 532-534 in File ez365.sol

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

✓ 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

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

Line 532-534 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 558-560 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 558-560 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 558-560 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 63

burn correctness

📅 09, Jul 2019
⏱ 189.43 ms

Line 551-557 in File ez365.sol

```
    /*@CTK "burn correctness"
      @tag assume_completion
      @post msg.sender != 0x0
      @post value <= _balances[msg.sender]
      @post __post._balances[msg.sender] == _balances[msg.sender] - value
      @post __post._totalSupply == _totalSupply - value
     */
```

Line 558-560 in File ez365.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 578-580 in File ez365.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 578-580 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 578-580 in File ez365.sol

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

✅ The code meets the specification.

Formal Verification Request 67

burnFrom correctness

📅 09, Jul 2019
⏱ 375.8 ms

Line 570-577 in File ez365.sol

```
    /*@CTK "burnFrom correctness"
      @tag assume_completion
      @post from != 0x0
      @post value <= _balances[from] && value <= _allowed[from][msg.sender]
      @post __post._balances[from] == _balances[from] - value
      @post __post._totalSupply == _totalSupply - value
      @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
     */
```

Line 578-580 in File ez365.sol

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

✅ 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

```
    //@CTK NO_OVERFLOW
```

Line 589-591 in File ez365.sol

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

✅ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 589-591 in File ez365.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 589-591 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 71

updateReleaseTokenTime correctness

09, Jul 2019
5.42 ms

Line 585-588 in File ez365.sol

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

Line 589-591 in File ez365.sol

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

✓ 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

```
    /*@CTK FAIL "transfer correctness"
      @tag assume_completion
      @post now >= _releaseTime || _owner == msg.sender
      @post _to != 0x0
      @post _value <= _balances[msg.sender]
      @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
           - _value
      @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
      @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
      @post __return == true
     */
```

Line 615-617 in File ez365.sol

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

This code violates the specification.

```
Counter Example:
Before 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": 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

```
  /*@CTK FAIL "transferFrom correctness"
    @tag assume_completion
    @post now >= _releaseTime || _owner == msg.sender
    @post _to != 0x0
    @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
    @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
    @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
    @post _to == _from -> __post._balances[_from] == _balances[_from]
    @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
    @post __return == true
   */
```

Line 633-635 in File ez365.sol

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

❌ This code violates the specification.

```
Counter Example:
Before 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": 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.

```
Counter Example:
Before 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"
    }
  ]

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

```
   //@CTK NO_OVERFLOW
```

Line 661-663 in File ez365.sol

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

✓ 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

```
   //@CTK NO_BUF_OVERFLOW
```

Line 661-663 in File ez365.sol

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

✓ 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

```
   //@CTK NO_ASF
```

Line 661-663 in File ez365.sol

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

✓ The code meets the specification.

Formal Verification Request 87

decreaseAllowance correctness

09, Jul 2019
3552.52 ms

Line 654-660 in File ez365.sol

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

Line 661-663 in File ez365.sol

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

This code violates the specification.

```
Counter Example:
Before 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"
      }
    ]

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

```
/**
 *Submitted for verification at Etherscan.io on 2019-04-12
*/

pragma solidity ^0.5.2;
/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    /**
     * @dev The Ownable constructor sets the original `owner` of the contract to the sender
     * account.
     */
    /*@CTK Ownable
      @post __post._owner == msg.sender
     */
    constructor () internal {
        _owner = msg.sender;
        emit OwnershipTransferred(address(0), _owner);
    }

    /**
     * @return the address of the owner.
     */
    /*@CTK owner
      @post __return == _owner
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(isOwner());
        _;
    }

    /**
     * @return true if `msg.sender` is the owner of the contract.
     */
    /*@CTK isOwner
      @post __return == (msg.sender == _owner)
     */
    function isOwner() public view returns (bool) {
        return msg.sender == _owner;
    }

    /**
     * @dev Allows the current owner to relinquish control of the contract.
     * It will not be possible to call the functions with the ‘onlyOwner‘
     * modifier anymore.
     * @notice Renouncing ownership will leave the contract without an owner,
     * thereby removing any functionality that is only available to the owner.
     */
    /*@CTK renounceOwnership
      @tag assume_completion
      @post _owner == msg.sender
      @post __post._owner == address(0)
     */
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    /**
     * @dev Allows the current owner to transfer control of the contract to a newOwner
         .
     * @param newOwner The address to transfer ownership to.
     */
    /*@CTK transferOwnership
      @tag assume_completion
      @post _owner == msg.sender
     */
    function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
    }

    /**
     * @dev Transfers control of the contract to a newOwner.
     * @param newOwner The address to transfer ownership to.
     */
    /*@CTK _transferOwnership
      @tag assume_completion
      @post newOwner != address(0)
      @post __post._owner == newOwner
     */
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0));
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}
/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error.
 */
library SafeMath {
    /**
     * @dev Multiplies two unsigned integers, reverts on overflow.
     */
    /*@CTK "SafeMath mul"
```

```
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
     * @dev Integer division of two unsigned integers truncating the quotient, reverts
          on division by zero.
     */
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
     * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend
         is greater than minuend).
     */
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Adds two unsigned integers, reverts on overflow.
     */
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> __return == a + b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
     * @dev Divides two unsigned integers and returns the remainder (unsigned integer
         modulo),
     * reverts when dividing by zero.
     */
    /*@CTK "SafeMath mod"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
/**
 * @title ERC20 interface
 * @dev see https://eips.ethereum.org/EIPS/eip-20
 */
interface IERC20 {
    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(address from, address to, uint256 value) external returns (
        bool);

    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender) external view returns (uint256)
        ;

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}

```

```
/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://eips.ethereum.org/EIPS/eip-20
 * Originally based on code by FirstBlood:
 * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.
    sol
 *
 * This implementation emits additional Approval events, allowing applications to
    reconstruct the allowance status for
 * all accounts just by listening to said events. Note that this isn't required by the
     specification, and other
 * compliant implementations may not do it.
 */
contract ERC20 is IERC20, Ownable {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowed;

    uint256 private _totalSupply;

    /**
     * @dev Total number of tokens in existence.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "totalSupply correctness"
      @post __return == _totalSupply
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev Gets the balance of the specified address.
     * @param owner The address to query the balance of.
     * @return A uint256 representing the amount owned by the passed address.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "balanceOf correctness"
      @post __return == _balances[owner]
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }

    /**
     * @dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param owner address The address which owns the funds.
     * @param spender address The address which will spend the funds.
     * @return A uint256 specifying the amount of tokens still available for the
        spender.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
      @post __return == _allowed[owner][spender]
     */
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token to a specified address.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "transfer correctness"
      @tag assume_completion
      @post to != 0x0
      @post value <= _balances[msg.sender]
      @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
          - value
      @post to != msg.sender -> __post._balances[to] == _balances[to] + value
      @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
      @post __return == true
     */
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on
         behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
          may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible
         solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
         desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param spender The address which will spend the funds.
     * @param value The amount of tokens to be spent.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve correctness"
      @pre msg.sender != 0x0
      @post spender == 0x0 -> __reverted
      @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
     */
    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another.
     * Note that while this function emits an Approval event, this is not required as
         per the specification,
     * and other compliant implementations may not emit the event.
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "transferFrom correctness"
      @tag assume_completion
      @post to != 0x0
      @post value <= _balances[from] && value <= _allowed[from][msg.sender]
      @post to != from -> __post._balances[from] == _balances[from] - value
      @post to != from -> __post._balances[to] == _balances[to] + value
      @post to == from -> __post._balances[from] == _balances[from]
      @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
      @post __return == true
     */
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when _allowed[msg.sender][spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param addedValue The amount of tokens to increase the allowance by.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "increaseAllowance correctness"
      @tag assume_completion
      @post spender != 0x0
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
      @post __return == true
    */
    function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
        return true;
    }

    /**
```

```
     * @dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param subtractedValue The amount of tokens to decrease the allowance by.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "decreaseAllowance correctness"
      @tag assume_completion
      @post spender != 0x0
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
          subtractedValue
      @post __return == true
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        return true;
    }

    /**
     * @dev Transfer token for a specified addresses.
     * @param from The address to transfer from.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    function _transfer(address from, address to, uint256 value) internal {
        require(to != address(0));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }

    /**
     * @dev Internal function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "_mint correctness"
      @tag assume_completion
      @post account != 0x0
      @post __post._balances[account] == _balances[account] + value
      @post __post._totalSupply == _totalSupply + value
     */
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    /**
     * @dev Approve an address to spend another addresses' tokens.
     * @param owner The address that owns the tokens.
     * @param spender The address that will spend the tokens.
     * @param value The number of tokens that can be spent.
     */
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * Emits an Approval event (reflecting the reduced allowance).
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
}


/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract ERC20Detailed is ERC20 {
    string constant private _name = "EZ365";
    string constant private _symbol = "EZ365";
    uint256 constant private _decimals = 18;


    /**
     * @return the name of the token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "ERC20Detailed name correctness"
      @post __return == _name
     */
    function name() public pure returns (string memory) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "ERC20Detailed symbol correctness"
      @post __return == _symbol
     */
    function symbol() public pure returns (string memory) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "ERC20Detailed decimals correctness"
      @post __return == _decimals
     */
    function decimals() public pure returns (uint256) {
        return _decimals;
    }
}
contract EZ365Token is ERC20Detailed {

    uint256 public _releaseTime;
    constructor() public {
        uint256 totalSupply = 200000000 * (10 ** decimals());
        _mint(msg.sender,totalSupply);
        _releaseTime = block.timestamp + 365 days;
    }
     /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of token to be burned.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "burn correctness"
```

```
      @tag assume_completion
      @post msg.sender != 0x0
      @post value <= _balances[msg.sender]
      @post __post._balances[msg.sender] == _balances[msg.sender] - value
      @post __post._totalSupply == _totalSupply - value
     */
    function burn(uint256 value) public {
        _burn(msg.sender, value);
    }

    /**
     * @dev Burns a specific amount of tokens from the target address and decrements
         allowance.
     * @param from address The account whose tokens will be burned.
     * @param value uint256 The amount of token to be burned.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "burnFrom correctness"
      @tag assume_completion
      @post from != 0x0
      @post value <= _balances[from] && value <= _allowed[from][msg.sender]
      @post __post._balances[from] == _balances[from] - value
      @post __post._totalSupply == _totalSupply - value
      @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
     */
    function burnFrom(address from, uint256 value) public {
        _burnFrom(from, value);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "updateReleaseTokenTime correctness"
      @post _owner != msg.sender -> __reverted
      @post _owner == msg.sender -> __post._releaseTime == tokenTime
     */
    function updateReleaseTokenTime(uint256 tokenTime) public onlyOwner {
        _releaseTime = tokenTime;
    }

    modifier isTokenReleased () {
        if (isOwner()){
           _;
        }else{
            require(block.timestamp >= _releaseTime);
            _;
        }
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "transfer correctness"
      @tag assume_completion
      @post now >= _releaseTime || _owner == msg.sender
      @post _to != 0x0
      @post _value <= _balances[msg.sender]
      @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
         - _value
      @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
      @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
      @post __return == true
     */
    function transfer(address _to, uint256 _value) public isTokenReleased returns (
        bool) {
        super.transfer(_to,_value);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "transferFrom correctness"
      @tag assume_completion
      @post now >= _releaseTime || _owner == msg.sender
      @post _to != 0x0
      @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
      @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
      @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
      @post _to == _from -> __post._balances[_from] == _balances[_from]
      @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
      @post __return == true
     */
   function transferFrom(address _from, address _to, uint256 _value) public
       isTokenReleased returns (bool) {
      super.transferFrom(_from, _to, _value);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "increaseAllowance correctness"
      @tag assume_completion
      @post now >= _releaseTime || _owner == msg.sender
      @post _spender != 0x0
      @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
          _addedValue
      @post __return == true
    */
   function increaseAllowance(address _spender, uint _addedValue) public
       isTokenReleased returns (bool) {
      super.increaseAllowance(_spender, _addedValue);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK FAIL "decreaseAllowance correctness"
      @tag assume_completion
      @post now >= _releaseTime || _owner == msg.sender
      @post _spender != 0x0
      @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] -
          _subtractedValue
      @post __return == true
     */
  function decreaseAllowance(address _spender, uint _subtractedValue) public
      isTokenReleased returns (bool) {
   super.decreaseAllowance(_spender, _subtractedValue);
  }
}
```