CertiK Audit Report For Leo

Request Date: 2019-05-16
Revision Date: 2019-05-21
Platform Name: Ethereum

Contents

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Leo(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

Exective Summary

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

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

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

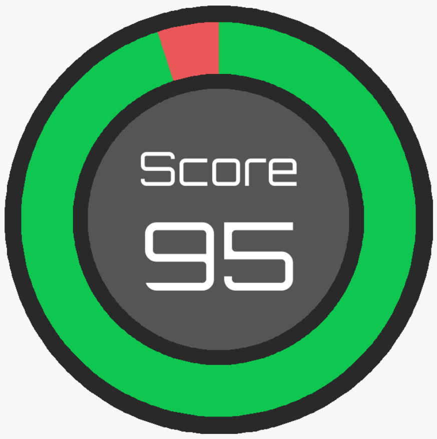
- Critical: The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.
- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The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

Testing Summary

PASS

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

May 21, 2019



Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

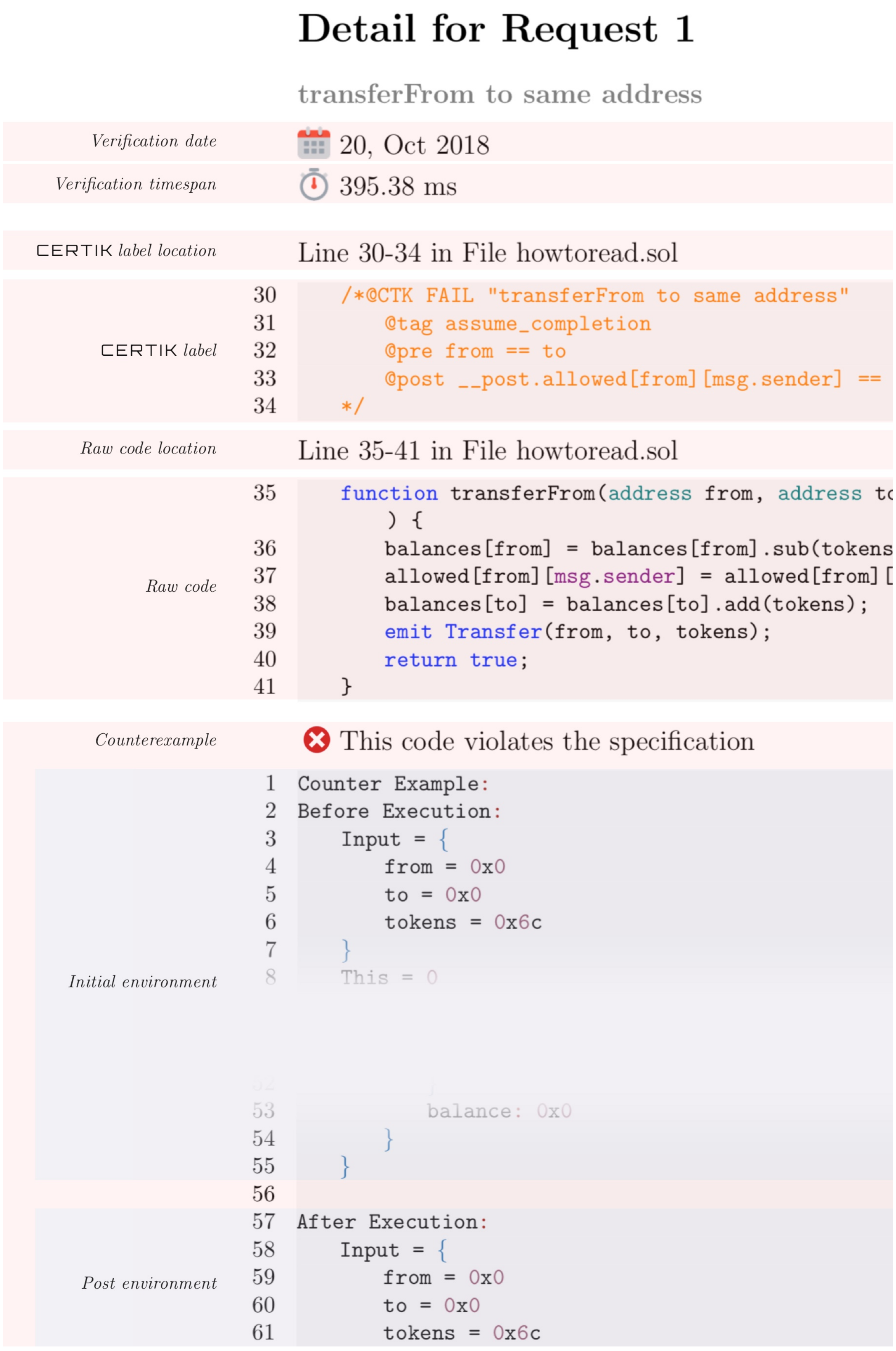
No issue found.

Low

Integer overflow issue that could be happening in function `getValueAt`, when `max` and `min` are really big. However, it is not very likely so we consider this to be of low priority.

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 t
        ) {
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

MiniMeToken

📅 21, May 2019
⏱ 53.85 ms

Line 112-119 in File MiniMeToken.sol

```
/*@CTK MiniMeToken
  @post __post.name == _tokenName
  @post __post.decimals == _decimalUnits
  @post __post.symbol == _tokenSymbol
  @post __post.parentSnapShotBlock == _parentSnapShotBlock
  @post __post.transfersEnabled == _transfersEnabled
  @post __post.creationBlock == block.number
 */
```

Line 120-137 in File MiniMeToken.sol

```
constructor(
    address _tokenFactory,
    address payable _parentToken,
    uint _parentSnapShotBlock,
    string memory _tokenName,
    uint8 _decimalUnits,
    string memory _tokenSymbol,
    bool _transfersEnabled
) public {
    // tokenFactory = MiniMeTokenFactory(_tokenFactory);
    name = _tokenName;                                 // Set the name
    decimals = _decimalUnits;                          // Set the decimals
    symbol = _tokenSymbol;                             // Set the symbol
    // parentToken = MiniMeToken(_parentToken);
    parentSnapShotBlock = _parentSnapShotBlock;
    transfersEnabled = _transfersEnabled;
    creationBlock = block.number;
}
```

✓ The code meets the specification

Formal Verification Request 2

approve

📅 21, May 2019
⏱ 37.51 ms

Line 239-244 in File MiniMeToken.sol

```
/*@CTK approve
  @tag assume_completion
  @post transfersEnabled
  @post (_amount == 0) || (allowed[msg.sender][_spender] == 0)
  @post __post.allowed[msg.sender][_spender] == _amount
 */
```

Line 245-262 in File MiniMeToken.sol

```
    function approve(address _spender, uint256 _amount) public returns (bool success)
        {
        require(transfersEnabled);

        // To change the approve amount you first have to reduce the addresses`
        //  allowance to zero by calling `approve(_spender,0)` if it is not
        //  already 0 to mitigate the race condition described here:
        //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
        require((_amount == 0) || (allowed[msg.sender][_spender] == 0));

        // Alerts the token controller of the approve function call
        // if (isContract(controller)) {
        //     require(TokenController(controller).onApprove(msg.sender, _spender,
            _amount));
        // }

        allowed[msg.sender][_spender] = _amount;
        emit Approval(msg.sender, _spender, _amount);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 3

generateTokens

📅 21, May 2019
⏱ 584.05 ms

Line 406-409 in File MiniMeToken.sol

```
    /*@CTK generateTokens
      @tag assume_completion
      @post controller == msg.sender
     */
```

Line 410-420 in File MiniMeToken.sol

```
    function generateTokens(address _owner, uint _amount
    ) public onlyController returns (bool) {
        uint curTotalSupply = totalSupply();
        require(curTotalSupply + _amount >= curTotalSupply); // Check for overflow
        uint previousBalanceTo = balanceOf(_owner);
        require(previousBalanceTo + _amount >= previousBalanceTo); // Check for
            overflow
        updateValueAtNow(totalSupplyHistory, curTotalSupply + _amount);
        updateValueAtNow(balances[_owner], previousBalanceTo + _amount);
        emit Transfer(address(0), _owner, _amount);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 4

destroyTokens

📅 21, May 2019
⏱ 263.94 ms

Line 427-430 in File MiniMeToken.sol

```
    /*@CTK destroyTokens
      @tag assume_completion
      @post controller == msg.sender
     */
```

Line 431-441 in File MiniMeToken.sol

```
    function destroyTokens(address _owner, uint _amount
    ) onlyController public returns (bool) {
        uint curTotalSupply = totalSupply();
        require(curTotalSupply >= _amount);
        uint previousBalanceFrom = balanceOf(_owner);
        require(previousBalanceFrom >= _amount);
        updateValueAtNow(totalSupplyHistory, curTotalSupply - _amount);
        updateValueAtNow(balances[_owner], previousBalanceFrom - _amount);
        emit Transfer(_owner, address(0), _amount);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 5

enableTransfers

📅 21, May 2019
⏱ 17.64 ms

Line 450-454 in File MiniMeToken.sol

```
    /*@CTK enableTransfers
      @tag assume_completion
      @post msg.sender == controller
      @post __post.transfersEnabled == _transfersEnabled
     */
```

Line 455-457 in File MiniMeToken.sol

```
    function enableTransfers(bool _transfersEnabled) public onlyController {
        transfersEnabled = _transfersEnabled;
    }
```

✓ The code meets the specification

Formal Verification Request 6

getValueAt

📅 21, May 2019
⏱ 3.92 ms

Line 467-470 in File MiniMeToken.sol

```
/*@CTK getValueAt
  @pre checkpoints.length == 0
  @post __return == 0
 */
```

Line 481-507 in File MiniMeToken.sol

```
function getValueAt(Checkpoint[] storage checkpoints, uint _block
) view internal returns (uint) {
    if (checkpoints.length == 0) return 0;

    // Shortcut for the actual value
    if (_block >= checkpoints[checkpoints.length-1].fromBlock)
        return checkpoints[checkpoints.length-1].value;
    if (_block < checkpoints[0].fromBlock) return 0;

    // Binary search of the value in the array
    uint min = 0;
    uint max = checkpoints.length-1;
    uint mid = 0;
    /*@CTK getValueAt_forLoop
      @inv max > min || max <= min
      @post max <= min
     */
    while (max > min) {
        mid = (max + min + 1)/ 2;
        if (checkpoints[mid].fromBlock<=_block) {
            min = mid;
        } else {
            max = mid-1;
        }
    }
    return checkpoints[min].value;
}
```

✓ The code meets the specification

Formal Verification Request 7

getValueAt_Min

21, May 2019
4.57 ms

Line 471-475 in File MiniMeToken.sol

```
/*@CTK getValueAt_Min
  @pre checkpoints.length > 0 && _block < checkpoints[0].fromBlock &&
       _block < checkpoints[checkpoints.length-1].fromBlock
  @post __return == 0
 */
```

Line 481-507 in File MiniMeToken.sol

```
function getValueAt(Checkpoint[] storage checkpoints, uint _block
) view internal returns (uint) {
```

```
483         if (checkpoints.length == 0) return 0;
484 
485         // Shortcut for the actual value
486         if (_block >= checkpoints[checkpoints.length-1].fromBlock)
487             return checkpoints[checkpoints.length-1].value;
488         if (_block < checkpoints[0].fromBlock) return 0;
489 
490         // Binary search of the value in the array
491         uint min = 0;
492         uint max = checkpoints.length-1;
493         uint mid = 0;
494         /*@CTK getValueAt_forLoop
495           @inv max > min || max <= min
496           @post max <= min
497          */
498         while (max > min) {
499             mid = (max + min + 1)/ 2;
500             if (checkpoints[mid].fromBlock<=_block) {
501                 min = mid;
502             } else {
503                 max = mid-1;
504             }
505         }
506         return checkpoints[min].value;
507     }
```

The code meets the specification

Formal Verification Request 8

getValueAt_Max

21, May 2019
4.35 ms

Line 476-480 in File MiniMeToken.sol

```
476     /*@CTK getValueAt_Max
477       @pre checkpoints.length > 0 &&
478           _block >= checkpoints[checkpoints.length-1].fromBlock
479       @post __return == checkpoints[checkpoints.length-1].value
480      */
```

Line 481-507 in File MiniMeToken.sol

```
481     function getValueAt(Checkpoint[] storage checkpoints, uint _block
482     ) view internal returns (uint) {
483         if (checkpoints.length == 0) return 0;
484 
485         // Shortcut for the actual value
486         if (_block >= checkpoints[checkpoints.length-1].fromBlock)
487             return checkpoints[checkpoints.length-1].value;
488         if (_block < checkpoints[0].fromBlock) return 0;
489 
490         // Binary search of the value in the array
491         uint min = 0;
492         uint max = checkpoints.length-1;
493         uint mid = 0;
```

```
        /*@CTK getValueAt_forLoop
          @inv max > min || max <= min
          @post max <= min
         */
        while (max > min) {
            mid = (max + min + 1)/ 2;
            if (checkpoints[mid].fromBlock<=_block) {
                min = mid;
            } else {
                max = mid-1;
            }
        }
        return checkpoints[min].value;
    }
```

✓ The code meets the specification

Formal Verification Request 9

min

📅 21, May 2019
⏱ 8.74 ms

Line 539-542 in File MiniMeToken.sol

```
    /*@CTK min
      @post a < b -> __return == a
      @post a > b -> __return == b
     */
```

Line 543-545 in File MiniMeToken.sol

```
    function min(uint a, uint b) pure internal returns (uint) {
        return a < b ? a : b;
    }
```

✓ The code meets the specification

Formal Verification Request 10

getValueAt_forLoop__Generated

📅 21, May 2019
⏱ 22.47 ms

(Loop) Line 494-497 in File MiniMeToken.sol

```
        /*@CTK getValueAt_forLoop
          @inv max > min || max <= min
          @post max <= min
         */
```

(Loop) Line 494-505 in File MiniMeToken.sol

```
        /*@CTK getValueAt_forLoop
          @inv max > min || max <= min
          @post max <= min
         */
        while (max > min) {
            mid = (max + min + 1)/ 2;
            if (checkpoints[mid].fromBlock<=_block) {
                min = mid;
            } else {
                max = mid-1;
            }
        }
```

✓ The code meets the specification

Formal Verification Request 11

onTransfer

21, May 2019
6.69 ms

Line 30-32 in File LEOController.sol

```
    /*@CTK onTransfer
      @post __return
     */
```

Line 33-35 in File LEOController.sol

```
    function onTransfer(address _from, address _to, uint _amount) public returns(bool)
         {
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 12

onApprove

21, May 2019
5.59 ms

Line 43-45 in File LEOController.sol

```
    /*@CTK onApprove
      @post __return
     */
```

Line 46-50 in File LEOController.sol

```
    function onApprove(address _owner, address _spender, uint _amount) public
        returns(bool)
    {
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 13

proxyPayment

📅 21, May 2019
⏱ 4.37 ms

Line 52-54 in File LEOController.sol

```
    /*@CTK proxyPayment
      @post !allowed
     */
```

Line 55-57 in File LEOController.sol

```
    function proxyPayment(address _owner) public payable returns(bool allowed) {
        allowed = false;
    }
```

✓ The code meets the specification

Formal Verification Request 14

Ownable

📅 21, May 2019
⏱ 5.63 ms

Line 9-11 in File Ownable.sol

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

Line 12-14 in File Ownable.sol

```
  constructor() public {
    owner = msg.sender;
  }
```

✓ The code meets the specification

Formal Verification Request 15

transferOwnership

📅 21, May 2019
⏱ 24.69 ms

Line 21-25 in File Ownable.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @post msg.sender == owner
  @post __post.owner == newOwner
 */
```

Line 26-30 in File Ownable.sol

```
function transferOwnership(address newOwner) public onlyOwner {
  require(newOwner != address(0));
  emit OwnershipTransferred(owner, newOwner);
  owner = newOwner;
}
```

✓ The code meets the specification

Formal Verification Request 16

Migrations

📅 21, May 2019
⏱ 5.35 ms

Line 7-9 in File Migrations.sol

```
/*@CTK Migrations
  @post __post.owner == msg.sender
 */
```

Line 10-12 in File Migrations.sol

```
constructor() public {
  owner = msg.sender;
}
```

✓ The code meets the specification

Formal Verification Request 17

Controlled

📅 21, May 2019
⏱ 5.81 ms

Line 13-15 in File Controlled.sol

```
  /*@CTK Controlled
    @post __post.controller == msg.sender
   */
```

Line 16 in File Controlled.sol

```
  constructor() public { controller = msg.sender;}
```

✓ The code meets the specification

Formal Verification Request 18

changeController

21, May 2019
16.3 ms

Line 20-24 in File Controlled.sol

```
    /*@CTK changeController
      @tag assume_completion
      @post msg.sender == controller
      @post __post.controller == _newController
     */
```

Line 25-28 in File Controlled.sol

```
    function changeController(address _newController) public onlyController {
        emit ControlTransferred(controller, _newController);
        controller = _newController;
    }
```

✓ The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File MiniMeToken.sol

```
pragma solidity ^0.5.0;
```

Version to compile has the following bug: 0.5.0: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.1: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStora
0.5.2: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.3: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.4: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.5: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization, DoubleShiftSizeOverflow 0.5.6: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization 0.5.7: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries 0.5.8: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File LEOController.sol

```
pragma solidity ^0.5.0;
```

Version to compile has the following bug: 0.5.0: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.1: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStora
0.5.2: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.3: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.4: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.5: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization, DoubleShiftSizeOverflow 0.5.6: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization 0.5.7: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries 0.5.8: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
pragma solidity ^0.5.0;
```

Version to compile has the following bug: 0.5.0: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.1: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStora
0.5.2: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.3: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.4: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.5: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization, DoubleShiftSizeOverflow 0.5.6: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization 0.5.7: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries 0.5.8: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

```
pragma solidity >=0.4.21 <0.6.0;
```

Version to compile has the following bug: 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2 0.5.0: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.1: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.2: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.3: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.4: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.5: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStora

IncorrectByteInstructionOptimization, DoubleShiftSizeOverflow 0.5.6: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization 0.5.7: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries 0.5.8: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File Controlled.sol

```
pragma solidity ^0.5.0;
```

Version to compile has the following bug: 0.5.0: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.1: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStora
0.5.2: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.3: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.4: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage 0.5.5: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization, DoubleShiftSizeOverflow 0.5.6: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries, ABIEncoderV2PackedStorage, IncorrectByteInstructionOptimization 0.5.7: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor, IncorrectEventSignatureInLibraries 0.5.8: DynamicConstructorArgumentsClippedABIV2

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **Ownable.sol** `9d142205eb280f24b4411b6358ef7a1298b7ca277d046ae2d34b415bbf73bfdc`
- **Migrations.sol** `1c4e30fd3aa765cb0ee259a29dead71c1c99888dcc7157c25df3405802cf5b09`
- **Controlled.sol** `3e0c5187b6e25e3652a881b54960fe3337d9a20c5ba21069cd45fa3e23845786`
- **MiniMeToken.sol** `7da493cb5c391446e03f54b3c33e144aac23b9f1d485e3df2873ff55074058e6`
- **TokenController.sol** `3c245879479af407de2fe831e521ad7d4e89db67a39ecd225c1366fd7453b5e8`
- **LEO.sol** `321d4588bf04fce0b1b3a7d2f1c08125cd62c3f2795eaf3fd3f0d3f8207f1cef`
- **LEOController.sol** `e24f4b6f08a5764fa4ebe2b3400009b3efbd2d23778e2c69e2b45116ab4d4ae7`

Summary

The LEO team asked CertiK to conduct a security audit of the design and implementation of its to-be-released MiniMe-based smart contracts. In this comprehensive audit, the source code analysis was conducted through a variety of methods and tools, such as CertiK Formal Verification as well as manual review by smart contract experts. CertiK directly interfaced with client-side engineers to fix critical loopholes and address recommended design changes throughout the audit process. The LEO team provided timely enhancements to source code suggestions, as well as supportive feedback surrounding the business logics.

At the moment, the LEO team did not have testing and documentation repositories available for reference. CertiK recommends additional unit test coverage, along with documentation, to more thoroughly simulate potential use cases and functionalities for token holders, especially with respect to super admin privileges that may impact the LEO token's decentralized nature.

Overall, CertiK observed that the contract follows good practices, using a reasonable amount of upgrades on top of the MiniMe prototype to facilitate the requirements of latest Solidity compiler. For the core purposes of the token, it seems like a wise decision for the LEO team to base the token on MiniMe; the token can revert back to a snapshot that identifies the total balance of each token holder, preventing potential manipulations and attacks in the future (though it should be noted that this is a trade-off of decentralization). With the final update of source code and delivery of the audit report, CertiK concludes that the contract is not vulnerable to the classically-known anti-patterns or security issues at this time. It should be noted that this audit report is not an absolute guarantee of correctness or trustworthiness, and CertiK always recommends seeking multiple opinions, increased test coverage, and live sandbox deployments before a mainnet release.

Recommendations

Items in this section are classified as Low Vulnerability to the overall security of the smart contracts. As a result, the client is able to decide whether these suggested changes will be reflected in the final deployed version of source code. If the client chooses to update the code, a copy of history will be recorded for future reference.

MiniMeToken.sol

1. `minime` is GPLv3 licensed.
    - LEO: We are strong believers in open source, and LEO contract will certainly be open source.
    - **Conclusion**: Resolved.
2. Recommend removing unused variable `totalPledgedFeesHistory`.
    - LEO: Removed in latest commit.
    - **Conclusion**: Resolved.
3. Recommend changing the type of fromBlock and value of `struct Checkpoint` to `uint256` to avoid number overflow and `updateValueAtNow` accordingly.
    - LEO: Adjusted in latest commit.
    - **Conclusion**: Resolved.
4. OpenZeppelin's SafeMath for math operations [trivial].
    - LEO: Will consider, however all overflow concerns are currently addressed with equivalent checks.
    - **Conclusion**: Resolved.
5. `getValueAt`: declaring `uint mid;` outside the while loop to save gas.
    - LEO: Moved `mid` to be initialized outside loop in latest commit.
    - **Conclusion**: Resolved.

Controlled.sol

1. `changeController`: Add address check `require(_newController != address(0), new controller is a zero address);`.
    - LEO: One possible upgrade path for LEO may include removing a controller and transferring control to address(0). This would be for example after tokens have been burned and we wish to give token holders guarantees that there would be no future minting or upgrades.
    - **Conclusion**: The implementation met the intention. Resolved.
2. Add event: `event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);`.
    - LEO: Event added in latest commit.

- **Conclusion**: Resolved.

3. Given the high importance of LEO token, consider having `MultiController` and `MultiOwner` logic just in case the potential risk of wrong addresses provided for the future `transferX` function.

    - LEO: The intention is that the owner will be a Gnosis multisig wallet. Willing to consider alternatives though if you propose.
    - **Conclusion**: This is a more preferred solution to have a multisig behind the scene acting as owner. Resolved.

LEOController.sol

1. Add `transfersEnabled` to enable pausing.

    - LEO: I do not believe we need to be able to pause transfers. If ever needed we can upgrade the controller.
    - **Conclusion**: Client expect the current `LEOController` to have minimal functionalities at current stage. Resolved.

2. `proxyPayment` always returns false.

    - LEO: Not sure if there is a recommendation here, however the intention with this controller is that it rollsback if fallback function is called.
    - **Conclusion**: Same as above.

3. Add logic to `onTransfer`, `onApprove`, etc.

    - LEO: This again may be considered in a future controller upgrade.
    - **Conclusion**: Same as above.

Source Code with CertiK Labels

File MiniMeToken.sol

```
pragma solidity ^0.5.0;


// Modified 2019, Will Harborne

/*
    Copyright 2016, Jordi Baylina

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.
 */

/// @title MiniMeToken Contract
/// @author Jordi Baylina
/// @dev This token contract's goal is to make it easy for anyone to clone this
///  token using the token distribution at a given block, this will allow DAO's
///  and DApps to upgrade their features in a decentralized manner without
///  affecting the original token
/// @dev It is ERC20 compliant, but still needs to under go further testing.

import "./Controlled.sol";
import "./TokenController.sol";

contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 _amount, address _token, bytes
        memory _data) public;
}

/// @dev The actual token contract, the default controller is the msg.sender
///  that deploys the contract, so usually this token will be deployed by a
///  token controller contract, which Giveth will call a "Campaign"
/// @dev The actual token contract, the default controller is the msg.sender
///  that deploys the contract, so usually this token will be deployed by a
///  token controller contract, which Giveth will call a "Campaign"
contract MiniMeToken is Controlled {

    string public name;             //The Token's name: e.g. DigixDAO Tokens
    uint8 public decimals;          //Number of decimals of the smallest unit
    string public symbol;           //An identifier: e.g. REP
    string public version = '3.0.0'; //An arbitrary versioning scheme


    /// @dev `Checkpoint` is the structure that attaches a block number to a
    ///  given value, the block number attached is the one that last changed the
    ///  value
    struct Checkpoint {

        // ‘fromBlock‘ is the block number that the value was generated from
        uint256 fromBlock;

        // ‘value‘ is the amount of tokens at a specific block number
        uint256 value;
    }

    // ‘parentToken‘ is the Token address that was cloned to produce this token;
    //  it will be 0x0 for a token that was not cloned
    // MiniMeToken public parentToken;

    // ‘parentSnapShotBlock‘ is the block number from the Parent Token that was
    //  used to determine the initial distribution of the Clone Token
    uint public parentSnapShotBlock;

    // ‘creationBlock‘ is the block number that the Clone Token was created
    uint public creationBlock;

    // ‘balances‘ is the map that tracks the balance of each address, in this
    //  contract when the balance changes the block number that the change
    //  occurred is also included in the map
    mapping (address => Checkpoint[]) balances;

    // ‘allowed‘ tracks any extra transfer rights as in all ERC20 tokens
    mapping (address => mapping (address => uint256)) allowed;

    // Tracks the history of the ‘totalSupply‘ of the token
    Checkpoint[] totalSupplyHistory;

    // Flag that determines if the token is transferable or not.
    bool public transfersEnabled;

    // Tracks the history of the ‘pledgedFees‘ belonging to token holders
    Checkpoint[] totalPledgedFeesHistory; // in wei

    // The factory used to create new clone tokens
    // MiniMeTokenFactory public tokenFactory;

////////////////
// Constructor
////////////////

    /// @notice Constructor to create a MiniMeToken
    /// @param _tokenFactory The address of the MiniMeTokenFactory contract that
    ///  will create the Clone token contracts, the token factory needs to be
    ///  deployed first
    /// @param _parentToken Address of the parent token, set to 0x0 if it is a
    ///  new token
    /// @param _parentSnapShotBlock Block of the parent token that will
    ///  determine the initial distribution of the clone token, set to 0 if it
    ///  is a new token
    /// @param _tokenName Name of the new token
    /// @param _decimalUnits Number of decimals of the new token
    /// @param _tokenSymbol Token Symbol for the new token
    /// @param _transfersEnabled If true, tokens will be able to be transferred
    /*@CTK MiniMeToken
      @post __post.name == _tokenName
      @post __post.decimals == _decimalUnits
      @post __post.symbol == _tokenSymbol
      @post __post.parentSnapShotBlock == _parentSnapShotBlock
      @post __post.transfersEnabled == _transfersEnabled
      @post __post.creationBlock == block.number
     */
    constructor(
        address _tokenFactory,
        address payable _parentToken,
        uint _parentSnapShotBlock,
        string memory _tokenName,
        uint8 _decimalUnits,
        string memory _tokenSymbol,
        bool _transfersEnabled
    ) public {
        // tokenFactory = MiniMeTokenFactory(_tokenFactory);
        name = _tokenName;                                 // Set the name
        decimals = _decimalUnits;                          // Set the decimals
        symbol = _tokenSymbol;                             // Set the symbol
        // parentToken = MiniMeToken(_parentToken);
        parentSnapShotBlock = _parentSnapShotBlock;
        transfersEnabled = _transfersEnabled;
        creationBlock = block.number;
    }


///////////////////
// ERC20 Methods
///////////////////

    uint constant MAX_UINT = 2**256 - 1;

    /// @notice Send `_amount` tokens to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _amount The amount of tokens to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _amount) public returns (bool success) {
        require(transfersEnabled);
        doTransfer(msg.sender, _to, _amount);
        return true;
    }

    /// @notice Send `_amount` tokens to `_to` from `_from` on the condition it
    ///  is approved by `_from`
    /// @param _from The address holding the tokens being transferred
    /// @param _to The address of the recipient
    /// @param _amount The amount of tokens to be transferred
    /// @return True if the transfer was successful
    function transferFrom(address _from, address _to, uint256 _amount
    ) public returns (bool success) {

        // The controller of this contract can move tokens around at will,
        //  this is important to recognize! Confirm that you trust the
        //  controller of this contract, which in most situations should be
        //  another open source smart contract or 0x0
        if (msg.sender != controller) {
            require(transfersEnabled);

            // The standard ERC 20 transferFrom functionality
            if (allowed[_from][msg.sender] < MAX_UINT) {
                require(allowed[_from][msg.sender] >= _amount);
                allowed[_from][msg.sender] -= _amount;
            }
        }
        doTransfer(_from, _to, _amount);
        return true;
    }

    /// @dev This is the actual transfer function in the token contract, it can
    ///  only be called by other functions in this contract.
    /// @param _from The address holding the tokens being transferred
    /// @param _to The address of the recipient
    /// @param _amount The amount of tokens to be transferred
    /// @return True if the transfer was successful
    function doTransfer(address _from, address _to, uint _amount
    ) internal {

           if (_amount == 0) {
               emit Transfer(_from, _to, _amount); // Follow the spec to louch the
                   event when transfer 0
               return;
           }

           require(parentSnapShotBlock < block.number);

           // Do not allow transfer to 0x0 or the token contract itself
           require((_to != address(0)) && (_to != address(this)));

           // If the amount being transfered is more than the balance of the
           //  account the transfer throws
           uint256 previousBalanceFrom = balanceOfAt(_from, block.number);

           require(previousBalanceFrom >= _amount);

           // Alerts the token controller of the transfer
           if (isContract(controller)) {
               require(TokenController(controller).onTransfer(_from, _to, _amount));
           }

           // First update the balance array with the new value for the address
           //  sending the tokens
           updateValueAtNow(balances[_from], previousBalanceFrom - _amount);

           // Then update the balance array with the new value for the address
           //  receiving the tokens
           uint256 previousBalanceTo = balanceOfAt(_to, block.number);
           require(previousBalanceTo + _amount >= previousBalanceTo); // Check for
               overflow
           updateValueAtNow(balances[_to], previousBalanceTo + _amount);

           // An event to make the transfer easy to find on the blockchain
           emit Transfer(_from, _to, _amount);

    }

    /// @param _owner The address that's balance is being requested
    /// @return The balance of '_owner' at the current block
    function balanceOf(address _owner) public view returns (uint256 balance) {
        return balanceOfAt(_owner, block.number);
    }

    /// @notice 'msg.sender' approves '_spender' to spend '_amount' tokens on
    ///  its behalf. This is a modified version of the ERC20 approve function
    ///  to be a little bit safer
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _amount The amount of tokens to be approved for transfer
    /// @return True if the approval was successful
    /*@CTK approve
      @tag assume_completion
      @post transfersEnabled
      @post (_amount == 0) || (allowed[msg.sender][_spender] == 0)
      @post __post.allowed[msg.sender][_spender] == _amount
     */
    function approve(address _spender, uint256 _amount) public returns (bool success)
        {
        require(transfersEnabled);

        // To change the approve amount you first have to reduce the addresses'
        //  allowance to zero by calling 'approve(_spender,0)' if it is not
        //  already 0 to mitigate the race condition described here:
        //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
        require((_amount == 0) || (allowed[msg.sender][_spender] == 0));

        // Alerts the token controller of the approve function call
        // if (isContract(controller)) {
        //     require(TokenController(controller).onApprove(msg.sender, _spender,
            _amount));
        // }

        allowed[msg.sender][_spender] = _amount;
        emit Approval(msg.sender, _spender, _amount);
        return true;
    }

    /// @dev This function makes it easy to read the 'allowed[]' map
    /// @param _owner The address of the account that owns the token
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens of _owner that _spender is allowed
    ///  to spend
    /*CTK allowance
      @post remaining == allowed[_owner][_spender]
     */
    function allowance(address _owner, address _spender
    ) public view returns (uint256 remaining) {
        return allowed[_owner][_spender];
    }

    /// @notice 'msg.sender' approves '_spender' to send '_amount' tokens on
    ///  its behalf, and then a function is triggered in the contract that is
    ///  being approved, '_spender'. This allows users to use their tokens to
    ///  interact with contracts in one function call instead of two
    /// @param _spender The address of the contract able to transfer the tokens
    /// @param _amount The amount of tokens to be approved for transfer
    /// @return True if the function call was successful
    function approveAndCall(address _spender, uint256 _amount, bytes memory _extraData
    ) public returns (bool success) {
        require(approve(_spender, _amount));

        ApproveAndCallFallBack(_spender).receiveApproval(
            msg.sender,
            _amount,
            address(this),
            _extraData
        );

        return true;
    }

    /// @dev This function makes it easy to get the total number of tokens
    /// @return The total number of tokens
    function totalSupply() public view returns (uint) {
        return totalSupplyAt(block.number);
    }


////////////////
// Query balance and totalSupply in History
////////////////

    /// @dev Queries the balance of `_owner` at a specific `_blockNumber`
    /// @param _owner The address from which the balance will be retrieved
    /// @param _blockNumber The block number when the balance is queried
    /// @return The balance at `_blockNumber`
    function balanceOfAt(address _owner, uint _blockNumber) public view
        returns (uint) {

        // These next few lines are used when the balance of the token is
        //  requested before a check point was ever created for this token, it
        //  requires that the `parentToken.balanceOfAt` be queried at the
        //  genesis block for that token as this contains initial balance of
        //  this token
        if ((balances[_owner].length == 0)
            || (balances[_owner][0].fromBlock > _blockNumber)) {
            // if (address(parentToken) != address(0)) {
            //     return parentToken.balanceOfAt(_owner, min(_blockNumber,
                parentSnapShotBlock));
            // } else {
                // Has no parent
                return 0;
            // }

        // This will return the expected balance during normal situations
        } else {
            return getValueAt(balances[_owner], _blockNumber);
        }
    }

    /// @notice Total amount of tokens at a specific `_blockNumber`.
    /// @param _blockNumber The block number when the totalSupply is queried
    /// @return The total amount of tokens at `_blockNumber`
    function totalSupplyAt(uint _blockNumber) public view returns(uint) {

        // These next few lines are used when the totalSupply of the token is
        //  requested before a check point was ever created for this token, it
        //  requires that the ‘parentToken.totalSupplyAt‘ be queried at the
        //  genesis block for this token as that contains totalSupply of this
        //  token at this block number.
        if ((totalSupplyHistory.length == 0)
            || (totalSupplyHistory[0].fromBlock > _blockNumber)) {
            // if (address(parentToken) != address(0)) {
            //     return parentToken.totalSupplyAt(min(_blockNumber,
                parentSnapShotBlock));
            // } else {
                return 0;
            // }

        // This will return the expected totalSupply during normal situations
        } else {
            return getValueAt(totalSupplyHistory, _blockNumber);
        }
    }

////////////////
// Clone Token Method
////////////////

    /// @notice Creates a new clone token with the initial distribution being
    ///  this token at ‘_snapshotBlock‘
    /// @param _cloneTokenName Name of the clone token
    /// @param _cloneDecimalUnits Number of decimals of the smallest unit
    /// @param _cloneTokenSymbol Symbol of the clone token
    /// @param _snapshotBlock Block when the distribution of the parent token is
    ///  copied to set the initial distribution of the new clone token;
    ///  if the block is zero than the actual block, the current block is used
    /// @param _transfersEnabled True if transfers are allowed in the clone
    /// @return The address of the new MiniMeToken Contract
    function createCloneToken(
        string memory _cloneTokenName,
        uint8 _cloneDecimalUnits,
        string memory _cloneTokenSymbol,
        uint _snapshotBlock,
        bool _transfersEnabled
        ) public returns(address) {
        if (_snapshotBlock == 0) _snapshotBlock = block.number;
        // MiniMeToken cloneToken = tokenFactory.createCloneToken(
        //     address(this),
        //     _snapshotBlock,
        //     _cloneTokenName,
        //     _cloneDecimalUnits,
        //     _cloneTokenSymbol,
        //     _transfersEnabled
        //     );

        cloneToken.changeController(msg.sender);

        // An event to make the token easy to find on the blockchain
        emit NewCloneToken(address(cloneToken), _snapshotBlock);
        return address(cloneToken);
    }

///////////////
// Generate and destroy tokens
///////////////

    /// @notice Generates `_amount` tokens that are assigned to `_owner`
    /// @param _owner The address that will be assigned the new tokens
    /// @param _amount The quantity of tokens generated
    /// @return True if the tokens are generated correctly
    /*@CTK generateTokens
      @tag assume_completion
      @post controller == msg.sender
     */
    function generateTokens(address _owner, uint _amount
    ) public onlyController returns (bool) {
        uint curTotalSupply = totalSupply();
        require(curTotalSupply + _amount >= curTotalSupply); // Check for overflow
        uint previousBalanceTo = balanceOf(_owner);
        require(previousBalanceTo + _amount >= previousBalanceTo); // Check for
            overflow
        updateValueAtNow(totalSupplyHistory, curTotalSupply + _amount);
        updateValueAtNow(balances[_owner], previousBalanceTo + _amount);
        emit Transfer(address(0), _owner, _amount);
        return true;
    }


    /// @notice Burns `_amount` tokens from `_owner`
    /// @param _owner The address that will lose the tokens
    /// @param _amount The quantity of tokens to burn
    /// @return True if the tokens are burned correctly
    /*@CTK destroyTokens
      @tag assume_completion
      @post controller == msg.sender
     */
    function destroyTokens(address _owner, uint _amount
    ) onlyController public returns (bool) {
        uint curTotalSupply = totalSupply();
        require(curTotalSupply >= _amount);
        uint previousBalanceFrom = balanceOf(_owner);
        require(previousBalanceFrom >= _amount);
        updateValueAtNow(totalSupplyHistory, curTotalSupply - _amount);
        updateValueAtNow(balances[_owner], previousBalanceFrom - _amount);
        emit Transfer(_owner, address(0), _amount);
        return true;
    }

///////////////
// Enable tokens transfers
///////////////


    /// @notice Enables token holders to transfer their tokens freely if true
    /// @param _transfersEnabled True if transfers are allowed in the clone
    /*@CTK enableTransfers
      @tag assume_completion
      @post msg.sender == controller
      @post __post.transfersEnabled == _transfersEnabled
     */
    function enableTransfers(bool _transfersEnabled) public onlyController {
        transfersEnabled = _transfersEnabled;
    }

////////////////
// Internal helper functions to query and set a value in a snapshot array
////////////////

    /// @dev ‘getValueAt‘ retrieves the number of tokens at a given block number
    /// @param checkpoints The history of values being queried
    /// @param _block The block number to retrieve the value at
    /// @return The number of tokens being queried
    /*@CTK getValueAt
      @pre checkpoints.length == 0
      @post __return == 0
     */
    /*@CTK getValueAt_Min
      @pre checkpoints.length > 0 && _block < checkpoints[0].fromBlock &&
           _block < checkpoints[checkpoints.length-1].fromBlock
      @post __return == 0
     */
    /*@CTK getValueAt_Max
      @pre checkpoints.length > 0 &&
           _block >= checkpoints[checkpoints.length-1].fromBlock
      @post __return == checkpoints[checkpoints.length-1].value
     */
    function getValueAt(Checkpoint[] storage checkpoints, uint _block
    ) view internal returns (uint) {
        if (checkpoints.length == 0) return 0;

        // Shortcut for the actual value
        if (_block >= checkpoints[checkpoints.length-1].fromBlock)
            return checkpoints[checkpoints.length-1].value;
        if (_block < checkpoints[0].fromBlock) return 0;

        // Binary search of the value in the array
        uint min = 0;
        uint max = checkpoints.length-1;
        uint mid = 0;
        /*@CTK getValueAt_forLoop
          @inv max > min || max <= min
          @post max <= min
         */
        while (max > min) {
            mid = (max + min + 1)/ 2;
            if (checkpoints[mid].fromBlock<=_block) {
                min = mid;
            } else {
                max = mid-1;
            }
        }
        return checkpoints[min].value;
    }

    /// @dev ‘updateValueAtNow‘ used to update the ‘balances‘ map and the
    ///  ‘totalSupplyHistory‘
    /// @param checkpoints The history of data being updated
    /// @param _value The new number of tokens
    function updateValueAtNow(Checkpoint[] storage checkpoints, uint _value
    ) internal {
        if ((checkpoints.length == 0)
        || (checkpoints[checkpoints.length -1].fromBlock < block.number)) {
               Checkpoint storage newCheckPoint = checkpoints[ checkpoints.length++ ];
               newCheckPoint.fromBlock = uint256(block.number);
               newCheckPoint.value = uint256(_value);
           } else {
               Checkpoint storage oldCheckPoint = checkpoints[checkpoints.length-1];
               oldCheckPoint.value = uint256(_value);
           }
    }

    /// @dev Internal function to determine if an address is a contract
    /// @param _addr The address being queried
    /// @return True if `_addr` is a contract
    function isContract(address _addr) view internal returns(bool) {
        uint size;
        if (_addr == address(0)) return false;
        assembly {
            size := extcodesize(_addr)
        }
        return size>0;
    }

    /// @dev Helper function to return a min betwen the two uints
    /*@CTK min
      @post a < b -> __return == a
      @post a > b -> __return == b
     */
    function min(uint a, uint b) pure internal returns (uint) {
        return a < b ? a : b;
    }

    /// @notice The fallback function: If the contract's controller has not been
    ///  set to 0, then the `proxyPayment` method is called which relays the
    ///  ether and creates tokens as described in the token controller contract
    function () external payable {
        require(isContract(controller));
        require(TokenController(controller).proxyPayment.value(msg.value)(msg.sender));
    }


////////////////
// Events
////////////////
    event ClaimedTokens(address indexed _token, address indexed _controller, uint
         _amount);
    event Transfer(address indexed _from, address indexed _to, uint256 _amount);
    event NewCloneToken(address indexed _cloneToken, uint _snapshotBlock);
    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _amount
        );

```

```
}


///////////////
// MiniMeTokenFactory
///////////////

/// @dev This contract is used to generate clone contracts from a contract.
///  In solidity this is the way to create a contract from a contract of the
///  same class
contract MiniMeTokenFactory {

    /// @notice Update the DApp by creating a new token with new functionalities
    ///  the msg.sender becomes the controller of this clone token
    /// @param _parentToken Address of the token being cloned
    /// @param _snapshotBlock Block of the parent token that will
    ///  determine the initial distribution of the clone token
    /// @param _tokenName Name of the new token
    /// @param _decimalUnits Number of decimals of the new token
    /// @param _tokenSymbol Token Symbol for the new token
    /// @param _transfersEnabled If true, tokens will be able to be transferred
    /// @return The address of the new token contract
    function createCloneToken(
        address payable _parentToken,
        uint _snapshotBlock,
        string memory _tokenName,
        uint8 _decimalUnits,
        string memory _tokenSymbol,
        bool _transfersEnabled
    ) public returns (MiniMeToken) {
        MiniMeToken newToken = new MiniMeToken(
            address(this),
            _parentToken,
            _snapshotBlock,
            _tokenName,
            _decimalUnits,
            _tokenSymbol,
            _transfersEnabled
            );

        newToken.changeController(msg.sender);
        return newToken;
    }
  }
```

File LEOController.sol

```
pragma solidity ^0.5.0;

import "./TokenController.sol";
import "./LEO.sol";
import "./Ownable.sol";

contract LEOController is TokenController, Ownable {

    LEO public tokenContract; // The new token for this Campaign

    /// @param _tokenAddress Address of the token contract this contract controls

    constructor(
        address payable _tokenAddress
    ) public {
        tokenContract = LEO(_tokenAddress);     // The Deployed Token Contract
    }


////////////////
// TokenController interface
////////////////

    /// @notice Notifies the controller about a transfer.
    /// Transfers can only happen to whitelisted addresses
    /// @param _from The origin of the transfer
    /// @param _to The destination of the transfer
    /// @param _amount The amount of the transfer
    /// @return False if the controller does not authorize the transfer
    /*@CTK onTransfer
      @post __return
     */
    function onTransfer(address _from, address _to, uint _amount) public returns(bool)
         {
        return true;
    }

    /// @notice Notifies the controller about an approval, for this Campaign all
    ///  approvals are allowed by default and no extra notifications are needed
    /// @param _owner The address that calls `approve()`
    /// @param _spender The spender in the `approve()` call
    /// @param _amount The amount in the `approve()` call
    /// @return False if the controller does not authorize the approval
    /*@CTK onApprove
      @post __return
     */
    function onApprove(address _owner, address _spender, uint _amount) public
        returns(bool)
    {
        return true;
    }

    /*@CTK proxyPayment
      @post !allowed
     */
    function proxyPayment(address _owner) public payable returns(bool allowed) {
        allowed = false;
    }

    /// @notice `onlyOwner` can upgrade the controller contract
    /// @param _newControllerAddress The address that will have the token control
        logic
    function upgradeController(address _newControllerAddress) public onlyOwner {
        tokenContract.changeController(_newControllerAddress);
        emit UpgradedController(_newControllerAddress);
    }

    function burnTokens(uint _amount) public onlyOwner returns (bool) {
        tokenContract.destroyTokens(owner, _amount);
    }

    function issueTokens(uint _amount) public onlyOwner returns (bool) {
        tokenContract.generateTokens(owner, _amount);
    }


//////////
// Safety Methods
//////////

    /// @notice This method can be used by the owner to extract mistakenly
    ///  sent tokens to this contract.
    /// @param _token The address of the token contract that you want to recover
    function claimLostTokens(address payable _token) public onlyOwner {

        LEO token = LEO(_token);
        uint balance = token.balanceOf(address(this));
        token.transfer(owner, balance);
        emit ClaimedTokens(_token, owner, balance);
    }

////////////////
// Events
////////////////
    event ClaimedTokens(address indexed _token, address indexed _controller, uint
        _amount);

    event UpgradedController (address newAddress);

}
```

File Ownable.sol

```
pragma solidity ^0.5.0;

contract Ownable {

  address public owner;

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

  /*@CTK Ownable
    @post __post.owner == msg.sender
   */
  constructor() public {
    owner = msg.sender;
  }

  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /*@CTK transferOwnership
    @tag assume_completion
    @post msg.sender == owner
    @post __post.owner == newOwner
   */
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }

}
```

File Migrations.sol

```
pragma solidity >=0.4.21 <0.6.0;

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  /*@CTK Migrations
    @post __post.owner == msg.sender
   */
  constructor() public {
    owner = msg.sender;
  }

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }

  function upgrade(address new_address) public restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

File Controlled.sol

```
pragma solidity ^0.5.0;

contract Controlled {

    event ControlTransferred(address indexed previousControler, address indexed
        newController);

    /// @notice The address of the controller is the only address that can call
    ///  a function with this modifier
    modifier onlyController { require(msg.sender == controller); _; }

    address public controller;

    /*@CTK Controlled
      @post __post.controller == msg.sender
     */
    constructor() public { controller = msg.sender;}

    /// @notice Changes the controller of the contract
    /// @param _newController The new controller of the contract
    /*@CTK changeController
      @tag assume_completion
      @post msg.sender == controller
      @post __post.controller == _newController
     */
    function changeController(address _newController) public onlyController {
        emit ControlTransferred(controller, _newController);
        controller = _newController;
    }
}
```