CertiK Verification Report For SampleClient

Request Date: 2018-12-12
Revision Date: 2018-12-14

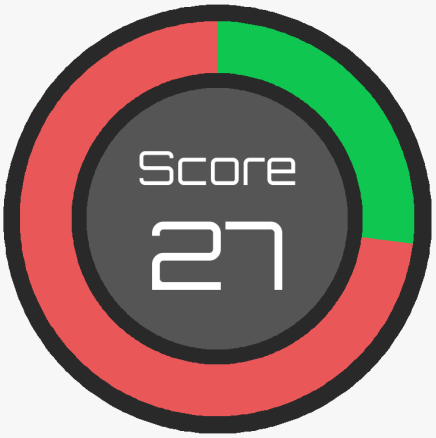
Disclaimer

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 SampleClient(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 CertiKs prior written consent.

WARNING

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

Dec 14, 2018



Summary

This is the report for smart contract verification service requestd by SampleClient. The goal of the audition is to guarantee that verified smart contracts are robust enough to avoid potentially unexpected loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the audit time.

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code by 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.	2	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	3	
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.	1	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.	1	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

Unlimited Burn:

- details

Unlimited Mint:

- details

TokenTransfer = Mint:

- details

Medium

No issue found.

Low

Insecure Compiler Version:

- details

Timestamp dependency:

- details

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.

Source Code with CertiK Labels

File sample.sol

```
pragma solidity ^0.4.19;

library SafeMath {
    /**
    * @dev Multiplies two numbers, throws on overflow.
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        if (a == 0) {
            return 0;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }

    /**
    * @dev Integer division of two numbers, truncating the quotient.
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // assert(b > 0); // Solidity automatically throws when dividing by 0
        // uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
        return a / b;
    }

    /**
    * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
        than minuend).
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    /**
    * @dev Adds two numbers, throws on overflow.
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
        c = a + b;
        assert(c >= a);
        return c;
    }
}

contract ExampleContract {
    using SafeMath for uint256;
    /**
     * Contract Administrator
     * @field status Contract external service status.
     * @field platformName Current contract platform name.
     * @field tokenSymbol token Symbol.
     * @field account Current contract administrator.
     * @field version Version of the contract.
```

```
54      */
55      struct Admin {
56          bool status;
57          bytes32 platformName;
58          bytes32 tokenSymbol;
59          address account;
60          string version;
61      }
62
63      Admin admin;
64      address[] private listOfPublicKey;
65      mapping (address => uint256) balances;
66      mapping (address => mapping (address => uint256)) public allowances;
67      uint256 public tokenMaxLimit;
68      uint256 public totalTokenSupply;
69
70      function ExampleContract(uint256 _tokenMaxLimit) {
71          admin.status = true;
72          admin.platformName = "ETH";
73          admin.tokenSymbol = "XXX";
74          admin.account = msg.sender;
75          admin.version = "1";
76          totalTokenSupply = 10 ** 25;
77          tokenMaxLimit = _tokenMaxLimit;
78      }
79
80      event Transfer(address indexed from, address indexed to, uint256 value);
81      event Burn(address indexed from, uint256 value);
82
83      //@CTK FAIL NO_OVERFLOW
84      //@CTK NO_BUF_OVERFLOW
85      //@CTK NO_ASF
86      /*@CTK "holder balance increases by the value"
87        @tag assume_completion
88        @post __post.balances[destination] == balances[destination] + amount
89        */
90      /*@CTK FAIL "total supply increases"
91        @tag assume_completion
92        @post __post.totalTokenSupply > totalTokenSupply
93        */
94      function mint(address destination, uint256 amount) external onlyAdmin {
95          require(amount != 0);
96          require(destination != address(0));
97          require(totalTokenSupply + amount <= tokenMaxLimit);
98
99          totalTokenSupply += amount;
100         balances[destination] += amount;
101         Transfer(0x0, destination, amount);
102     }
103
104     //@CTK FAIL NO_OVERFLOW
105     //@CTK NO_BUF_OVERFLOW
106     //@CTK NO_ASF
107     /*@CTK FAIL "If value is burned on other's behalf, allowance should be reduced."
108       @tag assume_completion
109       @pre success
110       @post __post.allowances[source_address][msg.sender] == allowances[source_address
              ][msg.sender] - burnAmount
```

```
111       */
112     function burnFrom(address source_address, uint256 burnAmount) returns (bool
            success) {
113         if (balances[source_address] < burnAmount) return false;
114         if (burnAmount > allowances[source_address][msg.sender]) return false;
115         totalTokenSupply -= burnAmount;
116         balances[source_address] -= burnAmount;
117         Burn(source_address, burnAmount);
118         return true;
119     }
120
121     modifier onlyAdmin {
122         require(admin.account == msg.sender);
123         _;
124     }
125
126     /**
127       * @dev Function to calculate the amount of vestable tokens at this moment.
128       * @param _to The address which will own the tokens.
129       * @return amount - A uint256 specifying the amount of tokens available to be
              vested at this moment.
130       * @return vestedMonths - A uint256 specifying the number of the vested months
              since the last vesting.
131       * @return curTime - A uint256 specifying the current timestamp.
132     */
133     struct TimeLock {
134         uint256 amount;      // total amount of tokens that is granted to the user
135         uint256 vestedAmount; // total amount of tokens that have been vested to the
              user
136         uint16 vestedMonths; // total number of months that
137         uint256 start;       // time when user is granted the tokens
138         uint256 cliff;       // time when user can start receive vested tokens
139         uint256 vesting;     // time when all the tokens can be vested
140         address from;        // token are granted to this user from address "from"
141     }
142     mapping(address => TimeLock) timeLocks;
143     uint256 constant MONTH = 3600*24*30;
144
145     /*@CTK FAIL "calculate vestable token between cliff and vesting"
146         @tag assume_completion
147         @pre now > timeLocks[_to].cliff && now <= timeLocks[_to].vesting
148         @pre (now - timeLocks[_to].start) / MONTH > timeLocks[_to].vestedMonths
149         @post vestedMonths == (now - timeLocks[_to].start) / MONTH - timeLocks[_to].
              vestedMonths
150         @post amount == timeLocks[_to].amount / ((timeLocks[_to].vesting - timeLocks[
              _to].start) / MONTH) * vestedMonths
151         @post curTime == now
152     */
153     function calcVestableToken(address _to)
154         internal view
155         returns (uint256 amount, uint256 vestedMonths, uint256 curTime)
156     {
157         uint256 vestTotalMonths;
158         uint256 vestedAmount;
159         uint256 vestPart;
160         amount = 0;
161         vestedMonths = 0;
162         curTime = now;
```

```
163
164         require(timeLocks[_to].amount > 0, "Nothing was granted to this address.");
165
166         if (curTime <= timeLocks[_to].cliff) {
167             return (0, 0, curTime);
168         }
169
170         vestedMonths = curTime.sub(timeLocks[_to].start) / MONTH;
171         vestedMonths = vestedMonths.sub(timeLocks[_to].vestedMonths);
172
173         if (curTime >= timeLocks[_to].vesting) {
174             return (timeLocks[_to].amount.sub(timeLocks[_to].vestedAmount),
                        vestedMonths, curTime);
175         }
176
177         if (vestedMonths > 0) {
178             vestTotalMonths = timeLocks[_to].vesting.sub(timeLocks[_to].start) / MONTH;
179             vestPart = timeLocks[_to].amount.div(vestTotalMonths);
180             amount = vestedMonths.mul(vestPart);
181             vestedAmount = timeLocks[_to].vestedAmount.add(amount);
182             if (vestedAmount > timeLocks[_to].amount) {
183                 vestedAmount = timeLocks[_to].amount.sub(timeLocks[_to].vestedAmount);
184             }
185         }
186
187         return (vestedAmount, vestedMonths, curTime);
188     }
189 }
```

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
```

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File sample.sol

```
1 pragma solidity ^0.4.19;
```

Only these compiler versions are safe to compile your code: 0.4.25

TIMESTAMP_DEPENDENCY

Line 162 in File sample.sol

```
162         curTime = now;
```

"now" can be influenced by minors to some degree

Formal Verification Request 1

If method completes, integer overflow would not happen.

14, Dec 2018
103.86 ms

Line 83 in File sample.sol

```
83      //@CTK FAIL NO_OVERFLOW
```

Line 94-102 in File sample.sol

```
94      function mint(address destination, uint256 amount) external onlyAdmin {
95          require(amount != 0);
96          require(destination != address(0));
97          require(totalTokenSupply + amount <= tokenMaxLimit);
98
99          totalTokenSupply += amount;
100         balances[destination] += amount;
101         Transfer(0x0, destination, amount);
102     }
```

This code violates the specification

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

```
            "tokenSymbol": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
            "account": 0,
            "version": ""
          },
          "listOfPublicKey": [],
          "balances": [
            {
              "key": 32,
              "value": 16
            },
            {
              "key": 1,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "tokenMaxLimit": 128,
          "totalTokenSupply": 0,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 128,
                "vestedAmount": 128,
                "vestedMonths": 128,
                "start": 128,
                "cliff": 128,
                "vesting": 128,
                "from": 128
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
    Input = {
```

```
        amount = 128
        destination = 1
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ExampleContract",
          "balance": 0,
          "contract": {
            "admin": {
              "status": false,
              "platformName": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
              "tokenSymbol": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
              "account": 0,
              "version": ""
            },
            "listOfPublicKey": [],
            "balances": [
              {
                "key": 32,
                "value": 16
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "allowances": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 0
                  }
                ]
              }
            ],
```

```
          "tokenMaxLimit": 128,
          "totalTokenSupply": 128,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 128,
                "vestedAmount": 128,
                "vestedMonths": 128,
                "start": 128,
                "cliff": 128,
                "vesting": 128,
                "from": 128
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 2

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

14, Dec 2018
20.52 ms

Line 84 in File sample.sol

```
84    //@CTK NO_BUF_OVERFLOW
```

Line 94-102 in File sample.sol

```
94    function mint(address destination, uint256 amount) external onlyAdmin {
95        require(amount != 0);
96        require(destination != address(0));
97        require(totalTokenSupply + amount <= tokenMaxLimit);
98
99        totalTokenSupply += amount;
100       balances[destination] += amount;
101       Transfer(0x0, destination, amount);
102   }
```

The code meets the specification

Formal Verification Request 3

Method will not encounter an assertion failure.

14, Dec 2018

⏱ 17.1 ms

Line 85 in File sample.sol

```
85    //@CTK NO_ASF
```

Line 94-102 in File sample.sol

```
94    function mint(address destination, uint256 amount) external onlyAdmin {
95        require(amount != 0);
96        require(destination != address(0));
97        require(totalTokenSupply + amount <= tokenMaxLimit);
98
99        totalTokenSupply += amount;
100       balances[destination] += amount;
101       Transfer(0x0, destination, amount);
102   }
```

✅ The code meets the specification

Formal Verification Request 4

holder balance increases by the value

📅 14, Dec 2018
⏱ 79.35 ms

Line 86-89 in File sample.sol

```
86    /*@CTK "holder balance increases by the value"
87      @tag assume_completion
88      @post __post.balances[destination] == balances[destination] + amount
89      */
```

Line 94-102 in File sample.sol

```
94    function mint(address destination, uint256 amount) external onlyAdmin {
95        require(amount != 0);
96        require(destination != address(0));
97        require(totalTokenSupply + amount <= tokenMaxLimit);
98
99        totalTokenSupply += amount;
100       balances[destination] += amount;
101       Transfer(0x0, destination, amount);
102   }
```

✅ The code meets the specification

Formal Verification Request 5

total supply increases

📅 14, Dec 2018
⏱ 176.58 ms

Line 90-93 in File sample.sol

```
90      /*@CTK FAIL "total supply increases"
91        @tag assume_completion
92        @post __post.totalTokenSupply > totalTokenSupply
93        */
```

Line 94-102 in File sample.sol

```
94      function mint(address destination, uint256 amount) external onlyAdmin {
95          require(amount != 0);
96          require(destination != address(0));
97          require(totalTokenSupply + amount <= tokenMaxLimit);
98
99          totalTokenSupply += amount;
100         balances[destination] += amount;
101         Transfer(0x0, destination, amount);
102     }
```

This code violates the specification

```
1   Counter Example:
2   Before Execution:
3       Input = {
4           amount = 1
5           destination = 64
6       }
7       This = 0
8       Internal = {
9           __has_assertion_failure = false
10          __has_buf_overflow = false
11          __has_overflow = false
12          __has_returned = false
13          __reverted = false
14          msg = {
15            "gas": 0,
16            "sender": 0,
17            "value": 0
18          }
19      }
20      Other = {
21          block = {
22            "number": 0,
23            "timestamp": 0
24          }
25      }
26      Address_Map = [
27        {
28          "key": 0,
29          "value": {
30            "contract_name": "ExampleContract",
31            "balance": 0,
32            "contract": {
33              "admin": {
34                "status": false,
35                "platformName": "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",
36                "tokenSymbol": "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",
37                "account": 0,
38                "version": ""
39              },
40              "listOfPublicKey": [],
41              "balances": [
```

```
            {
              "key": 0,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 64,
              "value": 255
            },
            {
              "key": 8,
              "value": 128
            },
            {
              "key": 96,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 255,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 1,
                "vestedAmount": 1,
                "vestedMonths": 1,
                "start": 1,
                "cliff": 1,
                "vesting": 1,
                "from": 1
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
```

```
        "value": "EmptyAddress"
      }
    ]

After Execution:
    Input = {
        amount = 1
        destination = 64
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ExampleContract",
          "balance": 0,
          "contract": {
            "admin": {
              "status": false,
              "platformName": "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",
              "tokenSymbol": "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",
              "account": 0,
              "version": ""
            },
            "listOfPublicKey": [],
            "balances": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 64,
                "value": 0
              },
              {
                "key": 8,
```

```
              "value": 128
            },
            {
              "key": 96,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 0,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 1,
                "vestedAmount": 1,
                "vestedMonths": 1,
                "start": 1,
                "cliff": 1,
                "vesting": 1,
                "from": 1
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```


Formal Verification Request 6

If method completes, integer overflow would not happen.

🗓 14, Dec 2018
⏱ 88.44 ms

Line 104 in File sample.sol

```
104     //@CTK FAIL NO_OVERFLOW
```

Line 112-119 in File sample.sol

```
112     function burnFrom(address source_address, uint256 burnAmount) returns (bool
            success) {
113         if (balances[source_address] < burnAmount) return false;
114         if (burnAmount > allowances[source_address][msg.sender]) return false;
115         totalTokenSupply -= burnAmount;
116         balances[source_address] -= burnAmount;
117         Burn(source_address, burnAmount);
118         return true;
119     }
```

This code violates the specification

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          burnAmount = 1
5          source_address = 0
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
12         __has_returned = false
13         __reverted = false
14         msg = {
15           "gas": 0,
16           "sender": 0,
17           "value": 0
18         }
19     }
20     Other = {
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25         success = false
26     }
27     Address_Map = [
28       {
29         "key": 0,
30         "value": {
31           "contract_name": "ExampleContract",
32           "balance": 0,
33           "contract": {
34             "admin": {
35               "status": false,
36               "platformName": "\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u
                     00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c
                     1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1",
37               "tokenSymbol": "\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u0
                     0c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1
                     \u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1",
38               "account": 0,
39               "version": ""
40             },
```

```
          "listOfPublicKey": [],
          "balances": [
            {
              "key": 1,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 4,
              "value": 8
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "allowances": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 0,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 128,
```

```
                  "vestedAmount": 128,
                  "vestedMonths": 128,
                  "start": 128,
                  "cliff": 128,
                  "vesting": 128,
                  "from": 128
                }
              }
            ],
            "MONTH": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

After Execution:
    Input = {
        burnAmount = 1
        source_address = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = true
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        success = true
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ExampleContract",
          "balance": 0,
          "contract": {
            "admin": {
              "status": false,
              "platformName": "\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u
                  00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c
                  1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1",
              "tokenSymbol": "\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u0
                  0c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1
```

```
                \u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1\u00c1",
            "account": 0,
            "version": ""
          },
          "listOfPublicKey": [],
          "balances": [
            {
              "key": 1,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 4,
              "value": 8
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 0,
              "value": 127
            },
            {
              "key": "ALL_OTHERS",
              "value": 128
            }
          ],
          "allowances": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            }
```

```
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 255,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 128,
                "vestedAmount": 128,
                "vestedMonths": 128,
                "start": 128,
                "cliff": 128,
                "vesting": 128,
                "from": 128
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 7

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

14, Dec 2018
0.7 ms

Line 105 in File sample.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 112-119 in File sample.sol

```
    function burnFrom(address source_address, uint256 burnAmount) returns (bool
        success) {
        if (balances[source_address] < burnAmount) return false;
        if (burnAmount > allowances[source_address][msg.sender]) return false;
        totalTokenSupply -= burnAmount;
        balances[source_address] -= burnAmount;
        Burn(source_address, burnAmount);
        return true;
    }
```

The code meets the specification

Formal Verification Request 8

Method will not encounter an assertion failure.

14, Dec 2018
0.68 ms

Line 106 in File sample.sol

```
106     //@CTK NO_ASF
```

Line 112-119 in File sample.sol

```
112     function burnFrom(address source_address, uint256 burnAmount) returns (bool
            success) {
113         if (balances[source_address] < burnAmount) return false;
114         if (burnAmount > allowances[source_address][msg.sender]) return false;
115         totalTokenSupply -= burnAmount;
116         balances[source_address] -= burnAmount;
117         Burn(source_address, burnAmount);
118         return true;
119     }
```

The code meets the specification

Formal Verification Request 9

If value is burned on other's behalf, allowance should be reduced.

14, Dec 2018
62.62 ms

Line 107-111 in File sample.sol

```
107     /*@CTK FAIL "If value is burned on other's behalf, allowance should be reduced."
108       @tag assume_completion
109       @pre success
110       @post __post.allowances[source_address][msg.sender] == allowances[source_address
              ][msg.sender] - burnAmount
111       */
```

Line 112-119 in File sample.sol

```
112     function burnFrom(address source_address, uint256 burnAmount) returns (bool
            success) {
113         if (balances[source_address] < burnAmount) return false;
114         if (burnAmount > allowances[source_address][msg.sender]) return false;
115         totalTokenSupply -= burnAmount;
116         balances[source_address] -= burnAmount;
117         Burn(source_address, burnAmount);
118         return true;
119     }
```

This code violates the specification

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

```
        burnAmount = 33
        source_address = 0
    }
    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 = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        success = false
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ExampleContract",
          "balance": 0,
          "contract": {
            "admin": {
              "status": false,
              "platformName": "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$",
              "tokenSymbol": "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$",
              "account": 0,
              "version": ""
            },
            "listOfPublicKey": [],
            "balances": [
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 10,
                "value": 2
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 4,
                "value": 0
```

```
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 0,
              "value": 64
            },
            {
              "key": "ALL_OTHERS",
              "value": 33
            }
          ],
          "allowances": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 33
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 4,
          "timeLocks": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 227,
                "vestedAmount": 227,
                "vestedMonths": 227,
                "start": 227,
                "cliff": 227,
                "vesting": 227,
                "from": 227
              }
            }
          ],
```

```
            "MONTH": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

After Execution:
    Input = {
        burnAmount = 33
        source_address = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = true
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        success = true
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "ExampleContract",
          "balance": 0,
          "contract": {
            "admin": {
              "status": false,
              "platformName": "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$",
              "tokenSymbol": "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$",
              "account": 0,
              "version": ""
            },
            "listOfPublicKey": [],
            "balances": [
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 10,
                "value": 2
```

```
            },
            {
              "key": 2,
              "value": 0
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": 4,
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 8,
              "value": 0
            },
            {
              "key": 0,
              "value": 31
            },
            {
              "key": "ALL_OTHERS",
              "value": 33
            }
          ],
          "allowances": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 64
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 33
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 227,
          "timeLocks": [
            {
```

```
              "key": "ALL_OTHERS",
              "value": {
                "amount": 227,
                "vestedAmount": 227,
                "vestedMonths": 227,
                "start": 227,
                "cliff": 227,
                "vesting": 227,
                "from": 227
              }
            }
          ],
          "MONTH": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 10

calculate vestable token between cliff and vesting

14, Dec 2018
1274.63 ms

Line 145-152 in File sample.sol

```
  /*@CTK FAIL "calculate vestable token between cliff and vesting"
      @tag assume_completion
      @pre now > timeLocks[_to].cliff && now <= timeLocks[_to].vesting
      @pre (now - timeLocks[_to].start) / MONTH > timeLocks[_to].vestedMonths
      @post vestedMonths == (now - timeLocks[_to].start) / MONTH - timeLocks[_to].
          vestedMonths
      @post amount == timeLocks[_to].amount / ((timeLocks[_to].vesting - timeLocks[
          _to].start) / MONTH) * vestedMonths
      @post curTime == now
  */
```

Line 153-188 in File sample.sol

```
  function calcVestableToken(address _to)
      internal view
      returns (uint256 amount, uint256 vestedMonths, uint256 curTime)
  {
      uint256 vestTotalMonths;
      uint256 vestedAmount;
      uint256 vestPart;
      amount = 0;
      vestedMonths = 0;
      curTime = now;

      require(timeLocks[_to].amount > 0, "Nothing was granted to this address.");

```

```
166         if (curTime <= timeLocks[_to].cliff) {
167             return (0, 0, curTime);
168         }
169 
170         vestedMonths = curTime.sub(timeLocks[_to].start) / MONTH;
171         vestedMonths = vestedMonths.sub(timeLocks[_to].vestedMonths);
172 
173         if (curTime >= timeLocks[_to].vesting) {
174             return (timeLocks[_to].amount.sub(timeLocks[_to].vestedAmount),
                        vestedMonths, curTime);
175         }
176 
177         if (vestedMonths > 0) {
178             vestTotalMonths = timeLocks[_to].vesting.sub(timeLocks[_to].start) / MONTH;
179             vestPart = timeLocks[_to].amount.div(vestTotalMonths);
180             amount = vestedMonths.mul(vestPart);
181             vestedAmount = timeLocks[_to].vestedAmount.add(amount);
182             if (vestedAmount > timeLocks[_to].amount) {
183                 vestedAmount = timeLocks[_to].amount.sub(timeLocks[_to].vestedAmount);
184             }
185         }
186 
187         return (vestedAmount, vestedMonths, curTime);
188     }
```

This code violates the specification

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _to = 0
5      }
6      This = 0
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         amount = 0
21         block = {
22           "number": 0,
23           "timestamp": 123
24         }
25         curTime = 0
26         vestedMonths = 0
27     }
28     Address_Map = [
29       {
30         "key": "ALL_OTHERS",
31         "value": {
32           "contract_name": "ExampleContract",
```

```
        "balance": 0,
        "contract": {
          "admin": {
            "status": false,
            "platformName": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
            "tokenSymbol": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
            "account": 0,
            "version": ""
          },
          "listOfPublicKey": [],
          "balances": [
            {
              "key": 0,
              "value": 4
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 0,
          "timeLocks": [
            {
              "key": 1,
              "value": {
                "amount": 0,
                "vestedAmount": 0,
                "vestedMonths": 0,
                "start": 0,
                "cliff": 0,
                "vesting": 0,
                "from": 0
              }
            },
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 3,
                "vestedAmount": 2,
                "vestedMonths": 0,
                "start": 119,
                "cliff": 0,
```

```
                "vesting": 123,
                "from": 0
              }
            }
          ],
          "MONTH": 1
        }
      }
    }
  ]

After Execution:
  Input = {
      _to = 0
  }
  This = 0
  Internal = {
      __has_assertion_failure = false
      __has_buf_overflow = false
      __has_overflow = false
      __has_returned = true
      __reverted = false
      msg = {
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      amount = 1
      block = {
        "number": 0,
        "timestamp": 123
      }
      curTime = 123
      vestPart = 0
      vestTotalMonths = 0
      vestedAmount = 0
      vestedMonths = 4
  }
  Address_Map = [
    {
      "key": "ALL_OTHERS",
      "value": {
        "contract_name": "ExampleContract",
        "balance": 0,
        "contract": {
          "admin": {
            "status": false,
            "platformName": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
            "tokenSymbol": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
            "account": 0,
            "version": ""
          },
          "listOfPublicKey": [],
          "balances": [
            {
              "key": 0,
```

```
              "value": 4
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "tokenMaxLimit": 0,
          "totalTokenSupply": 0,
          "timeLocks": [
            {
              "key": 1,
              "value": {
                "amount": 0,
                "vestedAmount": 0,
                "vestedMonths": 0,
                "start": 0,
                "cliff": 0,
                "vesting": 0,
                "from": 0
              }
            },
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 3,
                "vestedAmount": 2,
                "vestedMonths": 0,
                "start": 119,
                "cliff": 0,
                "vesting": 123,
                "from": 0
              }
            }
          ],
          "MONTH": 1
        }
      }
    }
  ]
```