CertiK Verification Report For Top Network

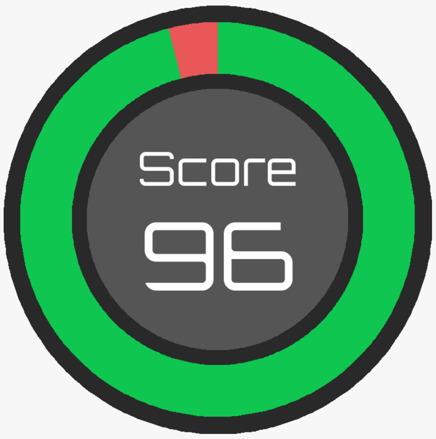
Request Date: 2019-03-19
Revision Date: 2019-03-19

Disclaimer

PASS

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

Mar 19, 2019



Summary

This audit report summarises the smart contract verification service requested by Top Network. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security 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 time of the audit.

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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

No issue found.

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.

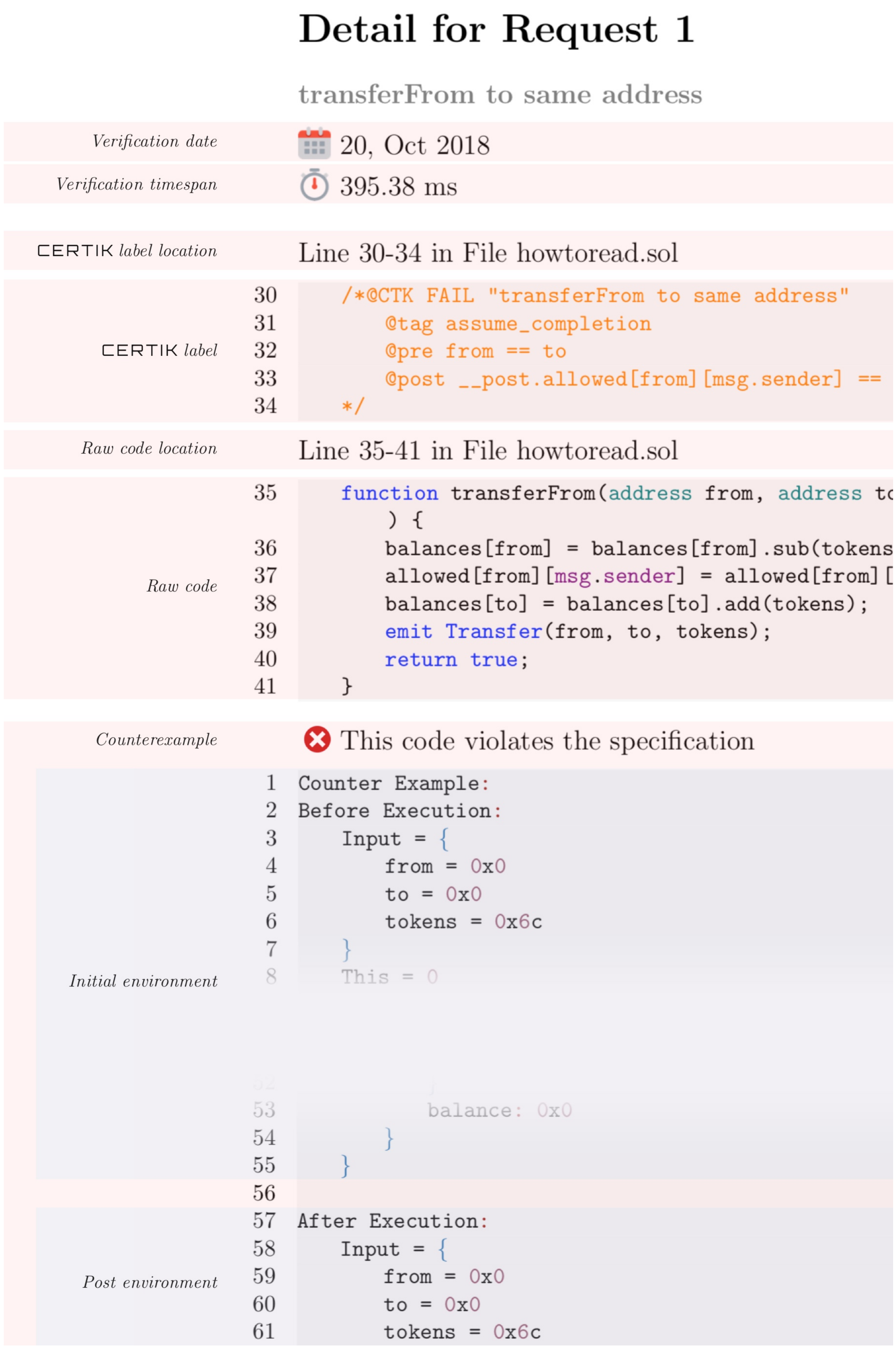
- Low: The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Source Code with CertiK Labels

File top.sol

```
pragma solidity ^0.5.6;

/**
 * @title ERC20Basic
 * @dev Simpler version of ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/179
 */
contract ERC20Basic {
  uint256 public totalSupply = 2e28;
  function balanceOf(address who) public view returns (uint256);
  function transfer(address to, uint256 value) public returns (bool);
  event Transfer(address indexed from, address indexed to, uint256 value);
}

contract TOPToken is ERC20Basic {
  bytes32 public name = "TOP Network";
  bytes32 public symbol = "TOP";
  uint256 public decimals = 18;
  address private owner = address(0);
  bool private active = false;

  mapping(address => uint256) private balances;

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

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  constructor() public {
    owner = msg.sender;
    balances[owner] = totalSupply;
    active = true;
  }

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

  /**
  * @dev transfer token for a specified address
  * @param _to The address to transfer to.
  * @param _value The amount to be transferred.
  */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "transfer2_same"
    @post (_to) == (msg.sender) -> (__reverted) == (true)
  */
  /*@CTK "transfer2"
    @tag assume_completion
    @pre (_to) != (msg.sender)
    @post (__post.balances[_to]) == ((balances[_to]) + (_value))
    @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_value))
    @post (__return) == (true)
  */
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(active);
    require(_to != address(0));
    require(_to != msg.sender);
    require(_value <= balances[msg.sender]);

    uint256 bal = balances[_to] + _value;
    require(bal >= balances[_to]);

    balances[msg.sender] = balances[msg.sender] - _value;
    balances[_to] = bal;

    emit Transfer(msg.sender, _to, _value);
    return true;
  }

  /**
  * @dev Gets the balance of the specified address.
  * @param _owner The address to query the the balance of.
  * @return An uint256 representing the amount owned by the passed address.
  */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "balanceOf"
    @pre (active) == (true)
    @post (__reverted) == (false)
    @post (bal) == (balances[_owner])
  */
  function balanceOf(address _owner) public view returns (uint256 bal) {
    require(active);
    return balances[_owner];
  }

  // Only owner can deactivate
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "deactivate"
    @post (msg.sender) != (owner) -> (__reverted) == (true)
    @post (msg.sender) == (owner) -> (__post.active) == (false)
  */
  function deactivate() public onlyOwner {
    active = false;
  }

  // Only owner can activate
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "activate"
    @post (msg.sender) != (owner) -> (__reverted) == (true)
    @post (msg.sender) == (owner) -> (__post.active) == (true)
  */
  function activate() public onlyOwner {
    active = true;
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "transferOwnership"
    @post (msg.sender) != (owner) -> (__reverted) == (true)
    @post ((msg.sender) == (owner) && (newOwner != 0x0))
          -> ((__post.owner) == (newOwner))
  */
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }

  // Only owner can kill
  function kill() public onlyOwner {
    require(!active);
    selfdestruct(msg.sender);
  }
}
```

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 tc
        ) {
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

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            }
            balance: 0x0
        }
    }

```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File top.sol

```
pragma solidity ^0.5.6;
```

Only these compiler versions are safe to compile your code: 0.5.6

Formal Verification Request 1

If method completes, integer overflow would not happen.

📅 19, Mar 2019
⏱ 24.86 ms

Line 26 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 29-33 in File top.sol

```
constructor() public {
  owner = msg.sender;
  balances[owner] = totalSupply;
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 2

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

📅 19, Mar 2019
⏱ 0.49 ms

Line 27 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 29-33 in File top.sol

```
constructor() public {
  owner = msg.sender;
  balances[owner] = totalSupply;
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 3

Method will not encounter an assertion failure.

📅 19, Mar 2019
⏱ 0.46 ms

Line 28 in File top.sol

```
//@CTK NO_ASF
```

Line 29-33 in File top.sol

```
constructor() public {
  owner = msg.sender;
  balances[owner] = totalSupply;
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 4

If method completes, integer overflow would not happen.

19, Mar 2019
119.89 ms

Line 45 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 58-72 in File top.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(active);
  require(_to != address(0));
  require(_to != msg.sender);
  require(_value <= balances[msg.sender]);

  uint256 bal = balances[_to] + _value;
  require(bal >= balances[_to]);

  balances[msg.sender] = balances[msg.sender] - _value;
  balances[_to] = bal;

  emit Transfer(msg.sender, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 5

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

19, Mar 2019
11.75 ms

Line 46 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 58-72 in File top.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(active);
  require(_to != address(0));
```

```
    require(_to != msg.sender);
    require(_value <= balances[msg.sender]);

    uint256 bal = balances[_to] + _value;
    require(bal >= balances[_to]);

    balances[msg.sender] = balances[msg.sender] - _value;
    balances[_to] = bal;

    emit Transfer(msg.sender, _to, _value);
    return true;
  }
```

✓ The code meets the specification

Formal Verification Request 6

Method will not encounter an assertion failure.

19, Mar 2019
11.81 ms

Line 47 in File top.sol

```
  //@CTK NO_ASF
```

Line 58-72 in File top.sol

```
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(active);
    require(_to != address(0));
    require(_to != msg.sender);
    require(_value <= balances[msg.sender]);

    uint256 bal = balances[_to] + _value;
    require(bal >= balances[_to]);

    balances[msg.sender] = balances[msg.sender] - _value;
    balances[_to] = bal;

    emit Transfer(msg.sender, _to, _value);
    return true;
  }
```

✓ The code meets the specification

Formal Verification Request 7

transfer2_same

19, Mar 2019
19.7 ms

Line 48-50 in File top.sol

```
/*@CTK "transfer2_same"
  @post (_to) == (msg.sender) -> (__reverted) == (true)
*/
```

Line 58-72 in File top.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(active);
  require(_to != address(0));
  require(_to != msg.sender);
  require(_value <= balances[msg.sender]);

  uint256 bal = balances[_to] + _value;
  require(bal >= balances[_to]);

  balances[msg.sender] = balances[msg.sender] - _value;
  balances[_to] = bal;

  emit Transfer(msg.sender, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 8

transfer2

19, Mar 2019
100.2 ms

Line 51-57 in File top.sol

```
/*@CTK "transfer2"
  @tag assume_completion
  @pre (_to) != (msg.sender)
  @post (__post.balances[_to]) == ((balances[_to]) + (_value))
  @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_value))
  @post (__return) == (true)
*/
```

Line 58-72 in File top.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(active);
  require(_to != address(0));
  require(_to != msg.sender);
  require(_value <= balances[msg.sender]);

  uint256 bal = balances[_to] + _value;
  require(bal >= balances[_to]);

  balances[msg.sender] = balances[msg.sender] - _value;
  balances[_to] = bal;

  emit Transfer(msg.sender, _to, _value);
  return true;
}
```

✓ The code meets the specification

Formal Verification Request 9

If method completes, integer overflow would not happen.

19, Mar 2019
19.08 ms

Line 79 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 87-90 in File top.sol

```
function balanceOf(address _owner) public view returns (uint256 bal) {
  require(active);
  return balances[_owner];
}
```

✓ The code meets the specification

Formal Verification Request 10

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

19, Mar 2019
0.54 ms

Line 80 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 87-90 in File top.sol

```
function balanceOf(address _owner) public view returns (uint256 bal) {
  require(active);
  return balances[_owner];
}
```

✓ The code meets the specification

Formal Verification Request 11

Method will not encounter an assertion failure.

19, Mar 2019
0.64 ms

Line 81 in File top.sol

```
//@CTK NO_ASF
```

Line 87-90 in File top.sol

```
function balanceOf(address _owner) public view returns (uint256 bal) {
  require(active);
  return balances[_owner];
}
```

✓ The code meets the specification

Formal Verification Request 12

balanceOf

19, Mar 2019
1.92 ms

Line 82-86 in File top.sol

```
/*@CTK "balanceOf"
  @pre (active) == (true)
  @post (__reverted) == (false)
  @post (bal) == (balances[_owner])
*/
```

Line 87-90 in File top.sol

```
function balanceOf(address _owner) public view returns (uint256 bal) {
  require(active);
  return balances[_owner];
}
```

✓ The code meets the specification

Formal Verification Request 13

If method completes, integer overflow would not happen.

19, Mar 2019
20.86 ms

Line 93 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 100-102 in File top.sol

```
function deactivate() public onlyOwner {
  active = false;
}
```

✓ The code meets the specification

Formal Verification Request 14

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

📅 19, Mar 2019
⏱ 0.51 ms

Line 94 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 100-102 in File top.sol

```
function deactivate() public onlyOwner {
  active = false;
}
```

✓ The code meets the specification

Formal Verification Request 15

Method will not encounter an assertion failure.

📅 19, Mar 2019
⏱ 0.5 ms

Line 95 in File top.sol

```
//@CTK NO_ASF
```

Line 100-102 in File top.sol

```
function deactivate() public onlyOwner {
  active = false;
}
```

✓ The code meets the specification

Formal Verification Request 16

deactivate

📅 19, Mar 2019
⏱ 1.83 ms

Line 96-99 in File top.sol

```
/*@CTK "deactivate"
  @post (msg.sender) != (owner) -> (__reverted) == (true)
  @post (msg.sender) == (owner) -> (__post.active) == (false)
*/
```

Line 100-102 in File top.sol

```
function deactivate() public onlyOwner {
  active = false;
}
```

✓ The code meets the specification

Formal Verification Request 17

If method completes, integer overflow would not happen.

19, Mar 2019
19.43 ms

Line 105 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 112-114 in File top.sol

```
function activate() public onlyOwner {
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 18

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

19, Mar 2019
0.51 ms

Line 106 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 112-114 in File top.sol

```
function activate() public onlyOwner {
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 19

Method will not encounter an assertion failure.

19, Mar 2019
0.5 ms

Line 107 in File top.sol

```
//@CTK NO_ASF
```

Line 112-114 in File top.sol

```
function activate() public onlyOwner {
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 20

activate

19, Mar 2019
1.68 ms

Line 108-111 in File top.sol

```
/*@CTK "activate"
  @post (msg.sender) != (owner) -> (__reverted) == (true)
  @post (msg.sender) == (owner) -> (__post.active) == (true)
*/
```

Line 112-114 in File top.sol

```
function activate() public onlyOwner {
  active = true;
}
```

✓ The code meets the specification

Formal Verification Request 21

If method completes, integer overflow would not happen.

19, Mar 2019
28.19 ms

Line 116 in File top.sol

```
//@CTK NO_OVERFLOW
```

Line 124-128 in File top.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 22

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

19, Mar 2019
0.62 ms

Line 117 in File top.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 124-128 in File top.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 23

Method will not encounter an assertion failure.

19, Mar 2019
0.63 ms

Line 118 in File top.sol

```
//@CTK NO_ASF
```

Line 124-128 in File top.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 24

transferOwnership

19, Mar 2019
3.17 ms

Line 119-123 in File top.sol

```
/*@CTK "transferOwnership"
  @post (msg.sender) != (owner) -> (__reverted) == (true)
  @post ((msg.sender) == (owner) && (newOwner != 0x0))
        -> ((__post.owner) == (newOwner))
*/
```

Line 124-128 in File top.sol

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

✓ The code meets the specification