CertiK Audit Report For LNX Protocol

Request Date: 2019-08-19
Revision Date: 2019-08-22
Platform Name: Ethereum

CERTIK

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

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.org/

Exective Summary

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

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

Vulnerability Classification

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

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

Low

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

Testing Summary

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.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **LNXProtocolToken.sol**[1]
  `296c8a2996d396c75c6ba130564891d0f0788a7886a04e49eec1bd7d79774901`

Summary

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

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments for the mainnet release.

[1]Etherscan Address: `https://etherscan.io/address/0x5e3845a1d78db544613edbe43dc1ea497266d3b8`

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 5 in File LNXProtocolToken.sol

```
5 pragma solidity ^0.5.9;
```

Only these compiler versions are safe to compile your code: 0.5.10

TIMESTAMP_DEPENDENCY

Line 340 in File LNXProtocolToken.sol

```
340         uint nowTime = now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 371 in File LNXProtocolToken.sol

```
371         uint nowTime = now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 405 in File LNXProtocolToken.sol

```
405         uint nowTime = now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 589 in File LNXProtocolToken.sol

```
589         uint nowTime = now;
```

"now" can be influenced by minors to some degree

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location	Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address to
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53         balance: 0x0
54     }
55 }
```

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

SafeMath mul

22, Aug 2019
85.78 ms

Line 9-14 in File LNXProtocolToken.sol

```
9   /*@CTK "SafeMath mul"
10    @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
11    @post !__reverted -> __return == a * b
12    @post !__reverted == !__has_overflow
13    @post !(__has_buf_overflow)
14   */
```

Line 15-21 in File LNXProtocolToken.sol

```
15  function mul(uint256 a, uint256 b) internal pure returns (uint256)
16    {
17  uint256 c = a * b;
18  assert(a == 0 || c / a == b);
19
20  return c;
21  }
```

The code meets the specification.

Formal Verification Request 2

SafeMath div

22, Aug 2019
6.2 ms

Line 23-29 in File LNXProtocolToken.sol

```
23  /*@CTK "SafeMath div"
24    @post (b == 0) == __reverted
25    @post !__reverted -> __return == a / b
26    @post !__reverted -> !__has_overflow
27    @post !__reverted -> !__has_assertion_failure
28    @post !(__has_buf_overflow)
29  */
```

Line 30-35 in File LNXProtocolToken.sol

```
30  function div(uint256 a, uint256 b) internal pure returns (uint256)
31 {
32  uint256 c = a / b;
33
34  return c;
35  }
```

The code meets the specification.

Formal Verification Request 3

SafeMath sub

22, Aug 2019
10.49 ms

Line 37-42 in File LNXProtocolToken.sol

```
37    /*@CTK "SafeMath sub"
38      @post (a < b) == __reverted
39      @post !__reverted -> __return == a - b
40      @post !__reverted -> !__has_overflow
41      @post !(__has_buf_overflow)
42     */
```

Line 43-48 in File LNXProtocolToken.sol

```
43    function sub(uint256 a, uint256 b) internal pure returns (uint256)
44  {
45    assert(b <= a);
46
47    return a - b;
48    }
```

The code meets the specification.

Formal Verification Request 4

SafeMath add

22, Aug 2019
12.77 ms

Line 50-55 in File LNXProtocolToken.sol

```
50    /*@CTK "SafeMath add"
51      @post (a + b < a || a + b < b) == __reverted
52      @post !__reverted -> __return == a + b
53      @post !__reverted -> !__has_overflow
54      @post !(__has_buf_overflow)
55     */
```

Line 56-62 in File LNXProtocolToken.sol

```
56    function add(uint256 a, uint256 b) internal pure returns (uint256)
57  {
58    uint256 c = a + b;
59    assert(c >= a);
60
61    return c;
62    }
```

The code meets the specification.

Formal Verification Request 5

OwnerHelper

22, Aug 2019
3.94 ms

Line 77-79 in File LNXProtocolToken.sol

```
77    /*@CTK OwnerHelper
78      @post __post.owner == msg.sender
79     */
```

Line 80-83 in File LNXProtocolToken.sol

```
80    constructor() public
81  {
82    owner = msg.sender;
83    }
```

The code meets the specification.

Formal Verification Request 6

transferOwnership

22, Aug 2019
29.36 ms

Line 84-89 in File LNXProtocolToken.sol

```
84    /*@CTK transferOwnership
85      @tag assume_completion
86      @post _to != owner
87      @post _to != address(0)
88      @post __post.owner == _to
89     */
```

Line 90-99 in File LNXProtocolToken.sol

```
90    function transferOwnership(address _to) onlyOwner public
91    {
92      require(_to != owner);
93      require(_to != address(0x0));
94
95        address from = owner;
96        owner = _to;
97
98        emit ChangeOwner(from, _to);
99    }
```

The code meets the specification.

Formal Verification Request 7

totalSupply

22, Aug 2019
4.81 ms

Line 242-244 in File LNXProtocolToken.sol

```
242 /*@CTK totalSupply
243   @post __return == totalTokenSupply
244  */
```

Line 245-248 in File LNXProtocolToken.sol

```
245 function totalSupply() view public returns (uint)
246 {
247     return totalTokenSupply;
248 }
```

The code meets the specification.

Formal Verification Request 8

balanceOf

22, Aug 2019
5.05 ms

Line 249-251 in File LNXProtocolToken.sol

```
249 /*@CTK balanceOf
250   @post __return == balances[_who]
251  */
```

Line 252-255 in File LNXProtocolToken.sol

```
252 function balanceOf(address _who) view public returns (uint)
253 {
254     return balances[_who];
255 }
```

The code meets the specification.

Formal Verification Request 9

approve

22, Aug 2019
67.64 ms

Line 275-280 in File LNXProtocolToken.sol

```
275 /*@CTK approve
276   @tag assume_completion
277   @post (msg.sender == owner) || !tokenLock
278   @post balances[msg.sender] >= _value
279   @post __post.approvals[msg.sender][_spender] == _value
280  */
```

Line 281-291 in File LNXProtocolToken.sol

```
function approve(address _spender, uint _value) public returns (bool)
{
    require(isTransferable() == true);
    require(balances[msg.sender] >= _value);

    approvals[msg.sender][_spender] = _value;

    emit Approval(msg.sender, _spender, _value);

    return true;
}
```

The code meets the specification.

Formal Verification Request 10

allowance

22, Aug 2019
5.49 ms

Line 292-294 in File LNXProtocolToken.sol

```
/*@CTK allowance
  @post __return == approvals[_owner][_spender]
 */
```

Line 295-298 in File LNXProtocolToken.sol

```
function allowance(address _owner, address _spender) view public returns (uint)
{
    return approvals[_owner][_spender];
}
```

The code meets the specification.

Formal Verification Request 11

transferFrom

22, Aug 2019
438.37 ms

Line 299-308 in File LNXProtocolToken.sol

```
/*@CTK transferFrom
  @tag assume_completion
  @pre _from != _to
  @post (msg.sender == owner) || !tokenLock
  @post balances[_from] >= _value
  @post approvals[_from][msg.sender] >= _value
  @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
      _value
  @post __post.balances[_from] == balances[_from] - _value
  @post __post.balances[_to] == balances[_to] + _value
 */
```

Line 309-322 in File LNXProtocolToken.sol

```
309 function transferFrom(address _from, address _to, uint _value) public returns (
        bool)
310 {
311     require(isTransferable() == true);
312     require(balances[_from] >= _value);
313     require(approvals[_from][msg.sender] >= _value);
314
315     approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
316     balances[_from] = balances[_from].sub(_value);
317     balances[_to] = balances[_to].add(_value);
318
319     emit Transfer(_from, _to, _value);
320
321     return true;
322 }
```

The code meets the specification.

Formal Verification Request 12

teamIssue

22, Aug 2019
1829.44 ms

Line 327-335 in File LNXProtocolToken.sol

```
327 /*@CTK teamIssue
328   @tag assume_completion
329   @post !saleTime
330   @post now > teamVestingTime
331   @post maxTeamSupply >= tokenIssuedTeam + teamVestingSupply
332   @post __post.balances[_to] == balances[_to] + teamVestingSupply
333   @post __post.totalTokenSupply == totalTokenSupply + teamVestingSupply
334   @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingSupply
335  */
```

Line 336-353 in File LNXProtocolToken.sol

```
336 function teamIssue(address _to) onlyOwner public
337 {
338     require(saleTime == false);
339
340     uint nowTime = now;
341     require(nowTime > teamVestingTime);
342
343     uint tokens = teamVestingSupply;
344
345     require(maxTeamSupply >= tokenIssuedTeam.add(tokens));
346
347     balances[_to] = balances[_to].add(tokens);
348
349     totalTokenSupply = totalTokenSupply.add(tokens);
350     tokenIssuedTeam = tokenIssuedTeam.add(tokens);
351
352     emit TeamIssue(_to, tokens);
353 }
```

The code meets the specification.

Formal Verification Request 13

rndIssue

22, Aug 2019
1425.92 ms

Line 354-365 in File LNXProtocolToken.sol

```
354   /*@CTK rndIssue
355     @tag assume_completion
356     @post !saleTime
357     @post _time < rndVestingTime
358     @post now > rndVestingTimer[_time]
359     @post rndVestingSupply == rndVestingBalances[_time]
360     @post maxRnDSupply >= tokenIssuedRnD + rndVestingSupply
361     @post __post.rndVestingBalances[_time] == 0
362     @post __post.balances[_to] == balances[_to] + rndVestingSupply
363     @post __post.totalTokenSupply == totalTokenSupply + rndVestingSupply
364     @post __post.tokenIssuedRnD == tokenIssuedRnD + rndVestingSupply
365    */
```

Line 366-386 in File LNXProtocolToken.sol

```
366   function rndIssue(address _to, uint _time) onlyOwner public
367   {
368       require(saleTime == false);
369       require(_time < rndVestingTime);
370
371       uint nowTime = now;
372       require( nowTime > rndVestingTimer[_time] );
373
374       uint tokens = rndVestingSupply;
375
376       require(tokens == rndVestingBalances[_time]);
377       require(maxRnDSupply >= tokenIssuedRnD.add(tokens));
378
379       balances[_to] = balances[_to].add(tokens);
380       rndVestingBalances[_time] = 0;
381
382       totalTokenSupply = totalTokenSupply.add(tokens);
383       tokenIssuedRnD = tokenIssuedRnD.add(tokens);
384
385       emit RnDIssue(_to, tokens);
386   }
```

The code meets the specification.

Formal Verification Request 14

advisorIssue

22, Aug 2019
1319.56 ms

Line 387-398 in File LNXProtocolToken.sol

```
/*@CTK advisorIssue
  @tag assume_completion
  @post !saleTime
  @post _time < advisorVestingTime
  @post now > advVestingTimer[_time]
  @post advisorVestingSupply == advVestingBalances[_time]
  @post maxAdvisorSupply >= tokenIssuedAdv + advisorVestingSupply
  @post __post.balances[_to] == balances[_to] + advisorVestingSupply
  @post __post.advVestingBalances[_time] == 0
  @post __post.totalTokenSupply == totalTokenSupply + advisorVestingSupply
  @post __post.tokenIssuedAdv == tokenIssuedAdv + advisorVestingSupply
 */
```

Line 400-420 in File LNXProtocolToken.sol

```
function advisorIssue(address _to, uint _time) onlyOwner public
{
    require(saleTime == false);
    require( _time < advisorVestingTime);

    uint nowTime = now;
    require( nowTime > advVestingTimer[_time] );

    uint tokens = advisorVestingSupply;

    require(tokens == advVestingBalances[_time]);
    require(maxAdvisorSupply >= tokenIssuedAdv.add(tokens));

    balances[_to] = balances[_to].add(tokens);
    advVestingBalances[_time] = 0;

    totalTokenSupply = totalTokenSupply.add(tokens);
    tokenIssuedAdv = tokenIssuedAdv.add(tokens);

    emit AdvIssue(_to, tokens);
}
```

The code meets the specification.

Formal Verification Request 15

ecoIssue

22, Aug 2019
438.13 ms

Line 421-429 in File LNXProtocolToken.sol

```
/*@CTK ecoIssue
  @tag assume_completion
  @post owner == msg.sender
  @post !saleTime
  @post tokenIssuedEco == 0
  @post __post.balances[_to] == balances[_to] + maxEcoSupply
  @post __post.totalTokenSupply == totalTokenSupply + maxEcoSupply
```

```
428     @post __post.tokenIssuedEco == tokenIssuedEco + maxEcoSupply
429    */
```

Line 430-443 in File LNXProtocolToken.sol

```
430    function ecoIssue(address _to) onlyOwner public
431    {
432        require(saleTime == false);
433        require(tokenIssuedEco == 0);
434
435        uint tokens = maxEcoSupply;
436
437        balances[_to] = balances[_to].add(tokens);
438
439        totalTokenSupply = totalTokenSupply.add(tokens);
440        tokenIssuedEco = tokenIssuedEco.add(tokens);
441
442        emit EcoIssue(_to, tokens);
443    }
```

The code meets the specification.

Formal Verification Request 16

mktIssue

22, Aug 2019
519.8 ms

Line 444-452 in File LNXProtocolToken.sol

```
444    /*@CTK mktIssue
445      @tag assume_completion
446      @post owner == msg.sender
447      @post !saleTime
448      @post tokenIssuedMkt == 0
449      @post __post.balances[_to] == balances[_to] + maxMktSupply
450      @post __post.totalTokenSupply == totalTokenSupply + maxMktSupply
451      @post __post.tokenIssuedMkt == tokenIssuedMkt + maxMktSupply
452     */
```

Line 453-466 in File LNXProtocolToken.sol

```
453    function mktIssue(address _to) onlyOwner public
454    {
455        require(saleTime == false);
456        require(tokenIssuedMkt == 0);
457
458        uint tokens = maxMktSupply;
459
460        balances[_to] = balances[_to].add(tokens);
461
462        totalTokenSupply = totalTokenSupply.add(tokens);
463        tokenIssuedMkt = tokenIssuedMkt.add(tokens);
464
465        emit EcoIssue(_to, tokens);
466    }
```

The code meets the specification.

Formal Verification Request 17

rsvIssue

22, Aug 2019
364.97 ms

Line 467-475 in File LNXProtocolToken.sol

```
467     /*@CTK rsvIssue
468       @tag assume_completion
469       @post msg.sender == owner
470       @post !saleTime
471       @post tokenIssuedRsv == 0
472       @post __post.balances[_to] == balances[_to] + maxReserveSupply
473       @post __post.totalTokenSupply == totalTokenSupply + maxReserveSupply
474       @post __post.tokenIssuedRsv == maxReserveSupply
475      */
```

Line 476-489 in File LNXProtocolToken.sol

```
476     function rsvIssue(address _to) onlyOwner public
477     {
478         require(saleTime == false);
479         require(tokenIssuedRsv == 0);
480
481         uint tokens = maxReserveSupply;
482
483         balances[_to] = balances[_to].add(tokens);
484
485         totalTokenSupply = totalTokenSupply.add(tokens);
486         tokenIssuedRsv = tokenIssuedRsv.add(tokens);
487
488         emit EcoIssue(_to, tokens);
489     }
```

The code meets the specification.

Formal Verification Request 18

privateSaleIssue

22, Aug 2019
451.72 ms

Line 490-497 in File LNXProtocolToken.sol

```
490     /*@CTK privateSaleIssue
491       @tag assume_completion
492       @post msg.sender == owner
493       @post tokenIssuedSale == 0
494       @post __post.balances[_to] == balances[_to] + privateSaleSupply
495       @post __post.totalTokenSupply == totalTokenSupply + privateSaleSupply
496       @post __post.tokenIssuedSale == tokenIssuedSale + privateSaleSupply
497      */
```

Line 498-510 in File LNXProtocolToken.sol

```
function privateSaleIssue(address _to) onlyOwner public
{
    require(tokenIssuedSale == 0);

    uint tokens = privateSaleSupply;

    balances[_to] = balances[_to].add(tokens);

    totalTokenSupply = totalTokenSupply.add(tokens);
    tokenIssuedSale = tokenIssuedSale.add(tokens);

    emit SaleIssue(_to, tokens);
}
```

The code meets the specification.

Formal Verification Request 19

publicSaleIssue

22, Aug 2019
642.29 ms

Line 511-518 in File LNXProtocolToken.sol

```
/*@CTK publicSaleIssue
  @tag assume_completion
  @post msg.sender == owner
  @post tokenIssuedSale == privateSaleSupply
  @post __post.balances[_to] == balances[_to] + publicSaleSupply
  @post __post.totalTokenSupply == totalTokenSupply + publicSaleSupply
  @post __post.tokenIssuedSale == tokenIssuedSale + publicSaleSupply
 */
```

Line 519-531 in File LNXProtocolToken.sol

```
function publicSaleIssue(address _to) onlyOwner public
{
    require(tokenIssuedSale == privateSaleSupply);

    uint tokens = publicSaleSupply;

    balances[_to] = balances[_to].add(tokens);

    totalTokenSupply = totalTokenSupply.add(tokens);
    tokenIssuedSale = tokenIssuedSale.add(tokens);

    emit SaleIssue(_to, tokens);
}
```

The code meets the specification.

Formal Verification Request 20

isTransferable

22, Aug 2019

3.26 ms

Line 536-538 in File LNXProtocolToken.sol

```
536 /*@CTK isTransferable
537   @post __return == !tokenLock || (msg.sender == owner)
538  */
```

Line 539-551 in File LNXProtocolToken.sol

```
539 function isTransferable() private view returns (bool)
540 {
541     if(tokenLock == false)
542     {
543         return true;
544     }
545     else if(msg.sender == owner)
546     {
547         return true;
548     }
549
550     return false;
551 }
```

The code meets the specification.

Formal Verification Request 21

setTokenUnlock

22, Aug 2019
43.84 ms

Line 552-557 in File LNXProtocolToken.sol

```
552 /*@CTK setTokenUnlock
553   @tag assume_completion
554   @post tokenLock
555   @post !saleTime
556   @post !__post.tokenLock
557  */
```

Line 558-564 in File LNXProtocolToken.sol

```
558 function setTokenUnlock() onlyOwner public
559 {
560     require(tokenLock == true);
561     require(saleTime == false);
562
563     tokenLock = false;
564 }
```

The code meets the specification.

Formal Verification Request 22

setTokenLock

22, Aug 2019
31.94 ms

Line 565-570 in File LNXProtocolToken.sol

```
565 /*@CTK setTokenLock
566   @tag assume_completion
567   @post owner == msg.sender
568   @post !tokenLock
569   @post __post.tokenLock
570  */
```

Line 571-576 in File LNXProtocolToken.sol

```
571 function setTokenLock() onlyOwner public
572 {
573     require(tokenLock == false);
574
575     tokenLock = true;
576 }
```

The code meets the specification.

Formal Verification Request 23

burnToken

22, Aug 2019
481.2 ms

Line 625-632 in File LNXProtocolToken.sol

```
625 /*@CTK burnToken
626   @tag assume_completion
627   @post owner == msg.sender
628   @post balances[msg.sender] >= _value * E18
629   @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
630   @post __post.burnTokenSupply == burnTokenSupply + _value * E18
631   @post __post.totalTokenSupply == totalTokenSupply - _value * E18
632  */
```

Line 633-645 in File LNXProtocolToken.sol

```
633 function burnToken(uint _value) onlyOwner public
634 {
635     uint tokens = _value * E18;
636
637     require(balances[msg.sender] >= tokens);
638
639     balances[msg.sender] = balances[msg.sender].sub(tokens);
640
641     burnTokenSupply = burnTokenSupply.add(tokens);
642     totalTokenSupply = totalTokenSupply.sub(tokens);
643
644     emit Burn(msg.sender, tokens);
```

```
}
```

The code meets the specification.

Source Code with CertiK Labels

File LNXProtocolToken.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-07-14
*/

pragma solidity ^0.5.9;

library SafeMath
{
    /*@CTK "SafeMath mul"
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256)
      {
    uint256 c = a * b;
    assert(a == 0 || c / a == b);

    return c;
    }

    /*@CTK "SafeMath div"
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256)
  {
    uint256 c = a / b;

    return c;
    }

    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256)
  {
    assert(b <= a);

    return a - b;
    }

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

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

    return c;
    }
}

contract OwnerHelper
{
    address public owner;

    event ChangeOwner(address indexed _from, address indexed _to);

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

    /*@CTK OwnerHelper
      @post __post.owner == msg.sender
     */
    constructor() public
  {
    owner = msg.sender;
    }
    /*@CTK transferOwnership
      @tag assume_completion
      @post _to != owner
      @post _to != address(0)
      @post __post.owner == _to
     */
    function transferOwnership(address _to) onlyOwner public
    {
      require(_to != owner);
      require(_to != address(0x0));

        address from = owner;
        owner = _to;

        emit ChangeOwner(from, _to);
    }
}

contract ERC20Interface
{
    event Transfer( address indexed _from, address indexed _to, uint _value);
    event Approval( address indexed _owner, address indexed _spender, uint _value);

    function totalSupply() view public returns (uint _supply);
    function balanceOf( address _who ) public view returns (uint _value);
    function transfer( address _to, uint _value) public returns (bool _success);
    function approve( address _spender, uint _value ) public returns (bool _success);
    function allowance( address _owner, address _spender ) public view returns (uint
        _allowance);
```

```
    function transferFrom( address _from, address _to, uint _value) public returns (
        bool _success);
}

contract LNXProtocolToken is ERC20Interface, OwnerHelper
{
    using SafeMath for uint;

    string public name;
    uint public decimals;
    string public symbol;

    uint constant private E18 = 1000000000000000000;
    uint constant private month = 2592000;

    // Total                                    2,473,750,000
    uint constant public maxTotalSupply =      2473750000 * E18;

    // Team                                      247,375,000 (10%)
    uint constant public maxTeamSupply =        247375000 * E18;

    // R&D                                       247,375,000 (10%)
    uint constant public maxRnDSupply =         247375000 * E18;

    // EcoSystem                                 371,062,500 (15%)
    uint constant public maxEcoSupply =         371062500 * E18;

    // Marketing                                 197,900,000 (8%)
    uint constant public maxMktSupply =         197900000 * E18;

    // Reserve                                   296,850,000 (12%)
    uint constant public maxReserveSupply =     296850000 * E18;

    // Advisor                                   123,687,500 (5%)
    uint constant public maxAdvisorSupply =     123687500 * E18;

    // Sale Supply                               989,500,000 (40%)
    uint constant public maxSaleSupply =        989500000 * E18;

    uint constant public publicSaleSupply =     100000000 * E18;
    uint constant public privateSaleSupply =    889500000 * E18;

    // Lock
    uint constant public rndVestingSupply      = 9895000 * E18;
    uint constant public rndVestingTime = 25;

    uint constant public teamVestingSupply     = 247375000 * E18;
    uint constant public teamVestingLockDate   = 24 * month;

    uint constant public advisorVestingSupply     = 30921875 * E18;
    uint constant public advisorVestingLockDate   = 3 * month;
    uint constant public advisorVestingTime = 4;

    uint public totalTokenSupply;
    uint public tokenIssuedTeam;
    uint public tokenIssuedRnD;
    uint public tokenIssuedEco;
    uint public tokenIssuedMkt;
```

```
    uint public tokenIssuedRsv;
    uint public tokenIssuedAdv;
    uint public tokenIssuedSale;

    uint public burnTokenSupply;

    mapping (address => uint) public balances;
    mapping (address => mapping ( address => uint )) public approvals;

    uint public teamVestingTime;

    mapping (uint => uint) public rndVestingTimer;
    mapping (uint => uint) public rndVestingBalances;

    mapping (uint => uint) public advVestingTimer;
    mapping (uint => uint) public advVestingBalances;

    bool public tokenLock = true;
    bool public saleTime = true;
    uint public endSaleTime = 0;

    event TeamIssue(address indexed _to, uint _tokens);
    event RnDIssue(address indexed _to, uint _tokens);
    event EcoIssue(address indexed _to, uint _tokens);
    event MktIssue(address indexed _to, uint _tokens);
    event RsvIssue(address indexed _to, uint _tokens);
    event AdvIssue(address indexed _to, uint _tokens);
    event SaleIssue(address indexed _to, uint _tokens);

    event Burn(address indexed _from, uint _tokens);

    event TokenUnlock(address indexed _to, uint _tokens);
    event EndSale(uint _date);

    /*CTK LNXProtocolToken
      @tag assume_completion
      @post __post.totalTokenSupply == 0
      @post __post.tokenIssuedTeam == 0
      @post __post.tokenIssuedRnD == 0
      @post __post.tokenIssuedEco == 0
      @post __post.tokenIssuedMkt == 0
      @post __post.tokenIssuedRsv == 0
      @post __post.tokenIssuedAdv == 0
      @post __post.tokenIssuedSale == 0
      @post __post.burnTokenSupply == 0
     */
    constructor() public
    {
        name       = "LNX Protocol";
        decimals   = 18;
        symbol     = "LNX";

        totalTokenSupply  = 0;

        tokenIssuedTeam = 0;
        tokenIssuedRnD    = 0;
        tokenIssuedEco   = 0;
        tokenIssuedMkt    = 0;
```

```
        tokenIssuedRsv   = 0;
        tokenIssuedAdv   = 0;
        tokenIssuedSale    = 0;

        burnTokenSupply    = 0;

        require(maxTeamSupply == teamVestingSupply);
        require(maxRnDSupply == rndVestingSupply.mul(rndVestingTime));
        require(maxAdvisorSupply == advisorVestingSupply.mul(advisorVestingTime));

        require(maxSaleSupply == publicSaleSupply + privateSaleSupply);
        require(maxTotalSupply == maxTeamSupply + maxRnDSupply + maxEcoSupply +
            maxMktSupply + maxReserveSupply + maxAdvisorSupply + maxSaleSupply);
    }

    // ERC - 20 Interface -----
    /*@CTK totalSupply
      @post __return == totalTokenSupply
     */
    function totalSupply() view public returns (uint)
    {
        return totalTokenSupply;
    }
    /*@CTK balanceOf
      @post __return == balances[_who]
     */
    function balanceOf(address _who) view public returns (uint)
    {
        return balances[_who];
    }
    /*CTK transfer
      @tag assume_completion
      @post (msg.sender == owner) || !tokenLock
      @post balances[msg.sender] >= _value
      @post __post.balances[msg.sender] == balances[msg.sender] - _value
      @post __post.balances[_to] == balances[_to] + _value
     */
    function transfer(address _to, uint _value) public returns (bool)
    {
        require(isTransferable() == true);
        require(balances[msg.sender] >= _value);

        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);

        emit Transfer(msg.sender, _to, _value);

        return true;
    }
    /*@CTK approve
      @tag assume_completion
      @post (msg.sender == owner) || !tokenLock
      @post balances[msg.sender] >= _value
      @post __post.approvals[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint _value) public returns (bool)
    {
        require(isTransferable() == true);
```

```
        require(balances[msg.sender] >= _value);

        approvals[msg.sender][_spender] = _value;

        emit Approval(msg.sender, _spender, _value);

        return true;
    }
    /*@CTK allowance
      @post __return == approvals[_owner][_spender]
     */
    function allowance(address _owner, address _spender) view public returns (uint)
    {
        return approvals[_owner][_spender];
    }
    /*@CTK transferFrom
      @tag assume_completion
      @pre _from != _to
      @post (msg.sender == owner) || !tokenLock
      @post balances[_from] >= _value
      @post approvals[_from][msg.sender] >= _value
      @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
          _value
      @post __post.balances[_from] == balances[_from] - _value
      @post __post.balances[_to] == balances[_to] + _value
     */
    function transferFrom(address _from, address _to, uint _value) public returns (
        bool)
    {
        require(isTransferable() == true);
        require(balances[_from] >= _value);
        require(approvals[_from][msg.sender] >= _value);

        approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);

        emit Transfer(_from, _to, _value);

        return true;
    }

    // -----

    // Vesting Function -----
    /*@CTK teamIssue
      @tag assume_completion
      @post !saleTime
      @post now > teamVestingTime
      @post maxTeamSupply >= tokenIssuedTeam + teamVestingSupply
      @post __post.balances[_to] == balances[_to] + teamVestingSupply
      @post __post.totalTokenSupply == totalTokenSupply + teamVestingSupply
      @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingSupply
     */
    function teamIssue(address _to) onlyOwner public
    {
        require(saleTime == false);

```

```
        uint nowTime = now;
        require(nowTime > teamVestingTime);

        uint tokens = teamVestingSupply;

        require(maxTeamSupply >= tokenIssuedTeam.add(tokens));

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedTeam = tokenIssuedTeam.add(tokens);

        emit TeamIssue(_to, tokens);
    }
    /*@CTK rndIssue
      @tag assume_completion
      @post !saleTime
      @post _time < rndVestingTime
      @post now > rndVestingTimer[_time]
      @post rndVestingSupply == rndVestingBalances[_time]
      @post maxRnDSupply >= tokenIssuedRnD + rndVestingSupply
      @post __post.rndVestingBalances[_time] == 0
      @post __post.balances[_to] == balances[_to] + rndVestingSupply
      @post __post.totalTokenSupply == totalTokenSupply + rndVestingSupply
      @post __post.tokenIssuedRnD == tokenIssuedRnD + rndVestingSupply
     */
    function rndIssue(address _to, uint _time) onlyOwner public
    {
        require(saleTime == false);
        require(_time < rndVestingTime);

        uint nowTime = now;
        require( nowTime > rndVestingTimer[_time] );

        uint tokens = rndVestingSupply;

        require(tokens == rndVestingBalances[_time]);
        require(maxRnDSupply >= tokenIssuedRnD.add(tokens));

        balances[_to] = balances[_to].add(tokens);
        rndVestingBalances[_time] = 0;

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedRnD = tokenIssuedRnD.add(tokens);

        emit RnDIssue(_to, tokens);
    }
    /*@CTK advisorIssue
      @tag assume_completion
      @post !saleTime
      @post _time < advisorVestingTime
      @post now > advVestingTimer[_time]
      @post advisorVestingSupply == advVestingBalances[_time]
      @post maxAdvisorSupply >= tokenIssuedAdv + advisorVestingSupply
      @post __post.balances[_to] == balances[_to] + advisorVestingSupply
      @post __post.advVestingBalances[_time] == 0
      @post __post.totalTokenSupply == totalTokenSupply + advisorVestingSupply
      @post __post.tokenIssuedAdv == tokenIssuedAdv + advisorVestingSupply
```

```
     */
    // _time : 0 ~ 3
    function advisorIssue(address _to, uint _time) onlyOwner public
    {
        require(saleTime == false);
        require( _time < advisorVestingTime);

        uint nowTime = now;
        require( nowTime > advVestingTimer[_time] );

        uint tokens = advisorVestingSupply;

        require(tokens == advVestingBalances[_time]);
        require(maxAdvisorSupply >= tokenIssuedAdv.add(tokens));

        balances[_to] = balances[_to].add(tokens);
        advVestingBalances[_time] = 0;

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedAdv = tokenIssuedAdv.add(tokens);

        emit AdvIssue(_to, tokens);
    }
    /*@CTK ecoIssue
      @tag assume_completion
      @post owner == msg.sender
      @post !saleTime
      @post tokenIssuedEco == 0
      @post __post.balances[_to] == balances[_to] + maxEcoSupply
      @post __post.totalTokenSupply == totalTokenSupply + maxEcoSupply
      @post __post.tokenIssuedEco == tokenIssuedEco + maxEcoSupply
     */
    function ecoIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedEco == 0);

        uint tokens = maxEcoSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedEco = tokenIssuedEco.add(tokens);

        emit EcoIssue(_to, tokens);
    }
    /*@CTK mktIssue
      @tag assume_completion
      @post owner == msg.sender
      @post !saleTime
      @post tokenIssuedMkt == 0
      @post __post.balances[_to] == balances[_to] + maxMktSupply
      @post __post.totalTokenSupply == totalTokenSupply + maxMktSupply
      @post __post.tokenIssuedMkt == tokenIssuedMkt + maxMktSupply
     */
    function mktIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
```

```
        require(tokenIssuedMkt == 0);

        uint tokens = maxMktSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedMkt = tokenIssuedMkt.add(tokens);

        emit EcoIssue(_to, tokens);
    }
    /*@CTK rsvIssue
      @tag assume_completion
      @post msg.sender == owner
      @post !saleTime
      @post tokenIssuedRsv == 0
      @post __post.balances[_to] == balances[_to] + maxReserveSupply
      @post __post.totalTokenSupply == totalTokenSupply + maxReserveSupply
      @post __post.tokenIssuedRsv == maxReserveSupply
     */
    function rsvIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedRsv == 0);

        uint tokens = maxReserveSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedRsv = tokenIssuedRsv.add(tokens);

        emit EcoIssue(_to, tokens);
    }
    /*@CTK privateSaleIssue
      @tag assume_completion
      @post msg.sender == owner
      @post tokenIssuedSale == 0
      @post __post.balances[_to] == balances[_to] + privateSaleSupply
      @post __post.totalTokenSupply == totalTokenSupply + privateSaleSupply
      @post __post.tokenIssuedSale == tokenIssuedSale + privateSaleSupply
     */
    function privateSaleIssue(address _to) onlyOwner public
    {
        require(tokenIssuedSale == 0);

        uint tokens = privateSaleSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedSale = tokenIssuedSale.add(tokens);

        emit SaleIssue(_to, tokens);
    }
    /*@CTK publicSaleIssue
      @tag assume_completion
      @post msg.sender == owner
```

```
     @post tokenIssuedSale == privateSaleSupply
     @post __post.balances[_to] == balances[_to] + publicSaleSupply
     @post __post.totalTokenSupply == totalTokenSupply + publicSaleSupply
     @post __post.tokenIssuedSale == tokenIssuedSale + publicSaleSupply
    */
   function publicSaleIssue(address _to) onlyOwner public
   {
       require(tokenIssuedSale == privateSaleSupply);

       uint tokens = publicSaleSupply;

       balances[_to] = balances[_to].add(tokens);

       totalTokenSupply = totalTokenSupply.add(tokens);
       tokenIssuedSale = tokenIssuedSale.add(tokens);

       emit SaleIssue(_to, tokens);
   }

   // -----

   // Lock Function -----
   /*@CTK isTransferable
     @post __return == !tokenLock || (msg.sender == owner)
    */
   function isTransferable() private view returns (bool)
   {
       if(tokenLock == false)
       {
           return true;
       }
       else if(msg.sender == owner)
       {
           return true;
       }

       return false;
   }
   /*@CTK setTokenUnlock
     @tag assume_completion
     @post tokenLock
     @post !saleTime
     @post !__post.tokenLock
    */
   function setTokenUnlock() onlyOwner public
   {
       require(tokenLock == true);
       require(saleTime == false);

       tokenLock = false;
   }
   /*@CTK setTokenLock
     @tag assume_completion
     @post owner == msg.sender
     @post !tokenLock
     @post __post.tokenLock
    */
   function setTokenLock() onlyOwner public
```

```
    {
        require(tokenLock == false);

        tokenLock = true;
    }

    // -----

    // ETC / Burn Function -----

    function endSale() onlyOwner public
    {
        require(saleTime == true);
        require(maxSaleSupply == tokenIssuedSale);

        saleTime = false;

        uint nowTime = now;
        endSaleTime = nowTime;

        teamVestingTime = endSaleTime + teamVestingLockDate;

        for(uint i = 0; i < rndVestingTime; i++)
        {
            rndVestingTimer[i] = endSaleTime + (month * i);
            rndVestingBalances[i] = rndVestingSupply;
        }

        for(uint i = 0; i < advisorVestingTime; i++)
        {
            advVestingTimer[i] = endSaleTime + (advisorVestingLockDate * i);
            advVestingBalances[i] = advisorVestingSupply;
        }

        emit EndSale(endSaleTime);
    }

    function withdrawTokens(address _contract, uint _decimals, uint _value) onlyOwner
        public
    {

        if(_contract == address(0x0))
        {
            uint eth = _value.mul(10 ** _decimals);
            msg.sender.transfer(eth);
        }
        else
        {
            uint tokens = _value.mul(10 ** _decimals);
            ERC20Interface(_contract).transfer(msg.sender, tokens);

            emit Transfer(address(0x0), msg.sender, tokens);
        }
    }
    /*@CTK burnToken
      @tag assume_completion
      @post owner == msg.sender
      @post balances[msg.sender] >= _value * E18
```

```
     @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
     @post __post.burnTokenSupply == burnTokenSupply + _value * E18
     @post __post.totalTokenSupply == totalTokenSupply - _value * E18
    */
   function burnToken(uint _value) onlyOwner public
   {
       uint tokens = _value * E18;

       require(balances[msg.sender] >= tokens);

       balances[msg.sender] = balances[msg.sender].sub(tokens);

       burnTokenSupply = burnTokenSupply.add(tokens);
       totalTokenSupply = totalTokenSupply.sub(tokens);

       emit Burn(msg.sender, tokens);
   }

   function close() onlyOwner public
   {
       selfdestruct(msg.sender);
   }

   // -----
}
```