

CERTIK AUDIT REPORT FOR VANTA



Request Date: 2019-05-10
Revision Date: 2019-05-12
Platform Name: Ethereum



Contents

Disclaimer	1
Exective Summary	2
Vulnerability Classification	2
Testing Summary	3
Audit Score	3
Type of Issues	3
Vulnerability Details	4
Formal Verification Results	5
How to read	5
Static Analysis Results	22
Manual Review Notes	23
Source Code with CertiK Labels	25

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 Vanta(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.

Executive Summary

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

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

Vulnerability Classification

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

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

Testing Summary

PASS

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

May 12, 2019



Type of Issues

CertiK smart label engine applied 100% covered 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.	4	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

No issue found.

Medium

No issue found.

Low



No issue found.

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
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c



Formal Verification Request 1

SafeMath_mul

 12, May 2019

 108.51 ms

Line 7-15 in File vanta.sol

```
7      /*@CTK SafeMath_mul
8      @tag spec
9      @post __reverted == __has_assertion_failure
10     @post __has_assertion_failure == __has_overflow
11     @post __reverted == false -> __return == a * b
12     @post msg == msg__post
13     @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
14     @post __addr_map == __addr_map__post
15     */
```

Line 16-22 in File vanta.sol


```
16     function mul(uint256 a, uint256 b) internal pure returns (uint256)
17     {
18         uint256 c = a * b;
19         assert(a == 0 || c / a == b);
20
21         return c;
22     }
```

✓ The code meets the specification

Formal Verification Request 2

SafeMath_div

 12, May 2019

 8.42 ms

Line 24-33 in File vanta.sol

```
24     /*@CTK SafeMath_div
25     @tag spec
26     @pre b != 0
27     @post __reverted == __has_assertion_failure
28     @post __has_overflow == true -> __has_assertion_failure == true
29     @post __reverted == false -> __return == a / b
30     @post msg == msg__post
31     @post (b == 0) == __has_assertion_failure
32     @post __addr_map == __addr_map__post
33     */
```

Line 34-39 in File vanta.sol

```
34     function div(uint256 a, uint256 b) internal pure returns (uint256)
35     {
36         uint256 c = a / b;
37
38         return c;
39     }
```

✓ The code meets the specification

Formal Verification Request 3

SafeMath_sub

📅 12, May 2019

🕒 13.86 ms

Line 41-49 in File vanta.sol

```
41  /*@CTK SafeMath_sub
42    @tag spec
43    @post __reverted == __has_assertion_failure
44    @post __has_overflow == true -> __has_assertion_failure == true
45    @post __reverted == false -> __return == a - b
46    @post msg == msg__post
47    @post (b > a) == __has_assertion_failure
48    @post __addr_map == __addr_map__post
49  */
```

Line 50-55 in File vanta.sol

```
50  function sub(uint256 a, uint256 b) internal pure returns (uint256)
51  {
52      assert(b <= a);
53
54      return a - b;
55  }
```

✓ The code meets the specification

Formal Verification Request 4

SafeMath_add

📅 12, May 2019

🕒 17.59 ms

Line 57-65 in File vanta.sol

```
57  /*@CTK SafeMath_add
58    @tag spec
59    @post __reverted == __has_assertion_failure
60    @post __has_assertion_failure == __has_overflow
61    @post __reverted == false -> __return == a + b
62    @post msg == msg__post
63    @post (a + b < a) == __has_assertion_failure
64    @post __addr_map == __addr_map__post
65  */
```

Line 66-72 in File vanta.sol

```
66  function add(uint256 a, uint256 b) internal pure returns (uint256)
67  {
68      uint256 c = a + b;
69      assert(c >= a);
```




```
70
71     return c;
72 }
```

✓ The code meets the specification

Formal Verification Request 5

OwnerHelper

 12, May 2019

 6.17 ms

Line 103-105 in File vanta.sol

```
103  /*@CTK OwnerHelper
104     @post __post.master == msg.sender
105  */
```

Line 106-109 in File vanta.sol


```
106  constructor() public
107  {
108      master = msg.sender;
109  }
```

✓ The code meets the specification

Formal Verification Request 6

transferMastership

 12, May 2019

 58.53 ms

Line 111-119 in File vanta.sol

```
111  /*@CTK transferMastership
112     @tag assume_completion
113     @post msg.sender == master
114     @post _to != master
115     @post _to != issuer
116     @post _to != manager
117     @post _to != address(0x0)
118     @post __post.master == _to
119  */
```

Line 120-131 in File vanta.sol

```
120  function transferMastership(address _to) onlyMaster public
121  {
122      require(_to != master);
123      require(_to != issuer);
124      require(_to != manager);
125      require(_to != address(0x0));
126
127      address from = master;
```

```

128         master = _to;
129
130         emit ChangeMaster(from, _to);
131     }

```

✓ The code meets the specification

Formal Verification Request 7

transferIssuer

📅 12, May 2019

🕒 57.97 ms

Line 133-141 in File vanta.sol

```

133     /*@CTK transferIssuer
134         @tag assume_completion
135         @post msg.sender == master
136         @post _to != master
137         @post _to != issuer
138         @post _to != manager
139         @post _to != address(0x0)
140         @post __post.issuer == _to
141     */

```

Line 142-153 in File vanta.sol

```

142     function transferIssuer(address _to) onlyMaster public
143     {
144         require(_to != master);
145         require(_to != issuer);
146         require(_to != manager);
147         require(_to != address(0x0));
148
149         address from = issuer;
150         issuer = _to;
151
152         emit ChangeIssuer(from, _to);
153     }

```

✓ The code meets the specification

Formal Verification Request 8

transferManager

📅 12, May 2019

🕒 57.17 ms

Line 155-163 in File vanta.sol

```

155     /*@CTK transferManager
156         @tag assume_completion
157         @post msg.sender == master
158         @post _to != master

```

```

159     @post _to != issuer
160     @post _to != manager
161     @post _to != address(0x0)
162     @post __post.manager == _to
163     */

```

Line 164-175 in File vanta.sol

```

164     function transferManager(address _to) onlyMaster public
165     {
166         require(_to != master);
167         require(_to != issuer);
168         require(_to != manager);
169         require(_to != address(0x0));
170
171         address from = manager;
172         manager = _to;
173
174         emit ChangeManager(from, _to);
175     }

```

✓ The code meets the specification

Formal Verification Request 9

VantaToken

📅 12, May 2019

🕒 2016.49 ms

Line 259-275 in File vanta.sol

```

259     /*@CTK VantaToken
260     @tag assume_completion
261     @post __post.name == "VANTA Token"
262     @post __post.decimals == 18
263     @post __post.symbol == "VNT"
264     @post __post.totalTokenSupply == 0
265     @post __post.tokenIssuedSale == 0
266     @post __post.tokenIssuedBdev == 0
267     @post __post.tokenIssuedMkt == 0
268     @post __post.tokenIssuedRnd == 0
269     @post __post.tokenIssuedTeam == 0
270     @post __post.tokenIssuedReserve == 0
271     @post __post.tokenIssuedAdvisor == 0
272     @post maxTotalSupply == maxSaleSupply + maxBdevSupply + maxMktSupply +
        maxRndSupply + maxTeamSupply + maxReserveSupply + maxAdvisorSupply
273     @post maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime
274     @post maxAdvisorSupply == advisorVestingSupplyPerTime * advisorVestingTime
275     */

```

Line 276-296 in File vanta.sol

```

276     constructor() public
277     {
278         name      = "VANTA Token";
279         decimals  = 18;
280         symbol    = "VNT";

```

```

281
282     totalTokenSupply = 0;
283
284     tokenIssuedSale    = 0;
285     tokenIssuedBdev    = 0;
286     tokenIssuedMkt     = 0;
287     tokenIssuedRnd     = 0;
288     tokenIssuedTeam    = 0;
289     tokenIssuedReserve = 0;
290     tokenIssuedAdvisor = 0;
291
292     require(maxTotalSupply == maxSaleSupply + maxBdevSupply + maxMktSupply +
293             maxRndSupply + maxTeamSupply + maxReserveSupply + maxAdvisorSupply);
294     // CTK: No safemath?
295     require(maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime);
296     require(maxAdvisorSupply == advisorVestingSupplyPerTime * advisorVestingTime);
297 }

```

✓ The code meets the specification

Formal Verification Request 10

totalSupply

📅 12, May 2019

🕒 5.65 ms

Line 300-302 in File vanta.sol

```

300     /*@CTK totalSupply
301         @post __return == totalTokenSupply
302     */

```

Line 303-306 in File vanta.sol

```

303     function totalSupply() view public returns (uint)
304     {
305         return totalTokenSupply;
306     }

```

✓ The code meets the specification

Formal Verification Request 11

balanceOf

📅 12, May 2019

🕒 16.95 ms

Line 308-312 in File vanta.sol

```

308     /*@CTK balanceOf
309         @tag assume_completion
310         @post __return == balances[_who] + privateFirstWallet[_who] +
311             privateSecondWallet[_who]
312     */

```

Line 313-319 in File vanta.sol


```
313     function balanceOf(address _who) view public returns (uint)
314     {
315         uint balance = balances[_who];
316         balance = balance.add(privateFirstWallet[_who] + privateSecondWallet[_who]);
317
318         return balance;
319     }
```

✓ The code meets the specification

Formal Verification Request 12

transfer

 12, May 2019

 234.66 ms

Line 321-327 in File vanta.sol

```
321     /*@CTK transfer
322         @tag assume_completion
323         @pre msg.sender != _to
324         @post balances[msg.sender] >= _value
325         @post __post.balances[msg.sender] == balances[msg.sender] - _value
326         @post __post.balances[_to] == balances[_to] + _value
327     */
```

Line 328-339 in File vanta.sol


```
328     function transfer(address _to, uint _value) public returns (bool)
329     {
330         require(isTransferable() == true);
331         require(balances[msg.sender] >= _value);
332
333         balances[msg.sender] = balances[msg.sender].sub(_value);
334         balances[_to] = balances[_to].add(_value);
335
336         emit Transfer(msg.sender, _to, _value);
337
338         return true;
339     }
```

✓ The code meets the specification

Formal Verification Request 13

approve

 12, May 2019

 60.2 ms

Line 341-346 in File vanta.sol



```
341  /*@CTK approve
342     @tag assume_completion
343     @pre msg.sender != _spender
344     @post balances[msg.sender] >= _value
345     @post __post.approvals[msg.sender][_spender] == _value
346  */
```

Line 347-357 in File vanta.sol

```
347  function approve(address _spender, uint _value) public returns (bool)
348  {
349      require(isTransferable() == true);
350      require(balances[msg.sender] >= _value);
351
352      approvals[msg.sender][_spender] = _value;
353
354      emit Approval(msg.sender, _spender, _value);
355
356      return true;
357  }
```

✓ The code meets the specification

Formal Verification Request 14

allowance

📅 12, May 2019

🕒 5.91 ms

Line 359-361 in File vanta.sol

```
359  /*@CTK allowance
360     @post __return == approvals[_owner][_spender]
361  */
```

Line 362-365 in File vanta.sol

```
362  function allowance(address _owner, address _spender) view public returns (uint)
363  {
364      return approvals[_owner][_spender];
365  }
```

✓ The code meets the specification

Formal Verification Request 15

transferFrom

📅 12, May 2019

🕒 460.33 ms

Line 367-374 in File vanta.sol

```
367  /*@CTK transferFrom
368     @tag assume_completion
369     @pre _from != _to
```



```

370     @post balances[_from] >= _value
371     @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
        _value
372     @post __post.balances[_from] == balances[_from] - _value
373     @post __post.balances[_to] == balances[_to] + _value
374     */

```

Line 375-388 in File vanta.sol

```

375     function transferFrom(address _from, address _to, uint _value) public returns (
        bool)
376     {
377         require(isTransferable() == true);
378         require(balances[_from] >= _value);
379         require(approvals[_from][msg.sender] >= _value);
380
381         approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
382         balances[_from] = balances[_from].sub(_value);
383         balances[_to] = balances[_to].add(_value);
384
385         emit Transfer(_from, _to, _value);
386
387         return true;
388     }

```

✓ The code meets the specification

Formal Verification Request 16

privateIssue

📅 12, May 2019

🕒 1596.14 ms

Line 393-402 in File vanta.sol

```

393     /*@CTK privateIssue
394         @tag assume_completion
395         @post maxSaleSupply >= tokenIssuedSale + _value * E18
396         @post __post.balances[_to] == balances[_to] + _value * E18 * 435 / 1000
397         @post __post.privateFirstWallet[_to] == privateFirstWallet[_to] + _value * E18 *
            435 / 1000
398         @post __post.privateSecondWallet[_to] == privateSecondWallet[_to] + _value * E18
            * 435 / 1000
399         @post __post.totalTokenSupply == totalTokenSupply + _value * E18
400         @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
401         @post __post.privateIssuedSale == privateIssuedSale + _value * E18
402     */

```

Line 403-417 in File vanta.sol

```

403     function privateIssue(address _to, uint _value) onlyIssuer public
404     {
405         uint tokens = _value * E18;
406         require(maxSaleSupply >= tokenIssuedSale.add(tokens));
407
408         balances[_to] = balances[_to].add( tokens.mul(435)/1000 );
409         privateFirstWallet[_to] = privateFirstWallet[_to].add( tokens.mul(435)
            /1000 );

```

```

410     privateSecondWallet[_to]      = privateSecondWallet[_to].add( tokens.mul(130)
        /1000 );
411
412     totalTokenSupply = totalTokenSupply.add(tokens);
413     tokenIssuedSale = tokenIssuedSale.add(tokens);
414     privateIssuedSale = privateIssuedSale.add(tokens);
415
416     emit SaleIssue(_to, tokens);
417 }

```

✓ The code meets the specification

Formal Verification Request 17

publicIssue

📅 12, May 2019

🕒 1339.04 ms

Line 419-426 in File vanta.sol

```

419  /*@CTK publicIssue
420     @tag assume_completion
421     @post maxSaleSupply >= tokenIssuedSale + _value * E18
422     @post __post.balances[_to] == balances[_to] + _value * E18
423     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
424     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
425     @post __post.publicIssuedSale == publicIssuedSale + _value * E18
426  */

```

Line 427-439 in File vanta.sol

```

427  function publicIssue(address _to, uint _value) onlyIssuer public
428  {
429      uint tokens = _value * E18;
430      require(maxSaleSupply >= tokenIssuedSale.add(tokens));
431
432      balances[_to] = balances[_to].add(tokens);
433
434      totalTokenSupply = totalTokenSupply.add(tokens);
435      tokenIssuedSale = tokenIssuedSale.add(tokens);
436      publicIssuedSale = publicIssuedSale.add(tokens);
437
438      emit SaleIssue(_to, tokens);
439  }

```

✓ The code meets the specification

Formal Verification Request 18

bdevIssue

📅 12, May 2019

🕒 930.36 ms

Line 441-447 in File vanta.sol



```
441  /*@CTK bdevIssue
442    @tag assume_completion
443    @post maxBdevSupply >= tokenIssuedBdev + _value * E18
444    @post __post.balances[_to] == balances[_to] + _value * E18
445    @post __post.totalTokenSupply == totalTokenSupply + _value * E18
446    @post __post.tokenIssuedBdev == tokenIssuedBdev + _value * E18
447  */
```

Line 448-459 in File vanta.sol

```
448  function bdevIssue(address _to, uint _value) onlyIssuer public
449  {
450      uint tokens = _value * E18;
451      require(maxBdevSupply >= tokenIssuedBdev.add(tokens));
452
453      balances[_to] = balances[_to].add(tokens);
454
455      totalTokenSupply = totalTokenSupply.add(tokens);
456      tokenIssuedBdev = tokenIssuedBdev.add(tokens);
457
458      emit BdevIssue(_to, tokens);
459  }
```

✓ The code meets the specification

Formal Verification Request 19

mktIssue



12, May 2019



721.72 ms

Line 461-467 in File vanta.sol

```
461  /*@CTK mktIssue
462    @tag assume_completion
463    @post maxMktSupply >= tokenIssuedMkt + _value * E18
464    @post __post.balances[_to] == balances[_to] + _value * E18
465    @post __post.totalTokenSupply == totalTokenSupply + _value * E18
466    @post __post.tokenIssuedMkt == tokenIssuedMkt + _value * E18
467  */
```

Line 468-479 in File vanta.sol


```
468  function mktIssue(address _to, uint _value) onlyIssuer public
469  {
470      uint tokens = _value * E18;
471      require(maxMktSupply >= tokenIssuedMkt.add(tokens));
472
473      balances[_to] = balances[_to].add(tokens);
474
475      totalTokenSupply = totalTokenSupply.add(tokens);
476      tokenIssuedMkt = tokenIssuedMkt.add(tokens);
477
478      emit MktIssue(_to, tokens);
479  }
```

✓ The code meets the specification

Formal Verification Request 20

rndIssue

 12, May 2019

 663.48 ms

Line 481-487 in File vanta.sol

```
481  /*@CTK rndIssue
482    @tag assume_completion
483    @post maxRndSupply >= tokenIssuedRnd + _value * E18
484    @post __post.balances[_to] == balances[_to] + _value * E18
485    @post __post.totalTokenSupply == totalTokenSupply + _value * E18
486    @post __post.tokenIssuedRnd == tokenIssuedRnd + _value * E18
487  */
```

Line 488-499 in File vanta.sol


```
488  function rndIssue(address _to, uint _value) onlyIssuer public
489  {
490      uint tokens = _value * E18;
491      require(maxRndSupply >= tokenIssuedRnd.add(tokens));
492
493      balances[_to] = balances[_to].add(tokens);
494
495      totalTokenSupply = totalTokenSupply.add(tokens);
496      tokenIssuedRnd = tokenIssuedRnd.add(tokens);
497
498      emit RndIssue(_to, tokens);
499  }
```

 The code meets the specification

Formal Verification Request 21

reserveIssue

 12, May 2019

 923.0 ms

Line 501-507 in File vanta.sol

```
501  /*@CTK reserveIssue
502    @tag assume_completion
503    @post maxReserveSupply >= tokenIssuedReserve + _value * E18
504    @post __post.balances[_to] == balances[_to] + _value * E18
505    @post __post.totalTokenSupply == totalTokenSupply + _value * E18
506    @post __post.tokenIssuedReserve == tokenIssuedReserve + _value * E18
507  */
```

Line 508-519 in File vanta.sol

```
508  function reserveIssue(address _to, uint _value) onlyIssuer public
509  {
510      uint tokens = _value * E18;
511      require(maxReserveSupply >= tokenIssuedReserve.add(tokens));
512
513      balances[_to] = balances[_to].add(tokens);
```

```

514
515     totalTokenSupply = totalTokenSupply.add(tokens);
516     tokenIssuedReserve = tokenIssuedReserve.add(tokens);
517
518     emit ReserveIssue(_to, tokens);
519 }


```

✓ The code meets the specification

Formal Verification Request 22

teamIssueVesting

 12, May 2019

 8102.8 ms

Line 524-536 in File vanta.sol

```

524 /*@CTK teamIssueVesting
525    @tag assume_completion
526    @post msg.sender == issuer
527    @post !saleTime
528    @post teamVestingTime >= _time
529    @post (endSaleTime + _time * teamVestingDate < now) &&
530          (teamVestingTimeAtSupply[_time] > 0)
531    @post maxTeamSupply >= tokenIssuedTeam + teamVestingTimeAtSupply[_time]
532    @post __post.balances[_to] == balances[_to] + teamVestingTimeAtSupply[_time]
533    @post __post.teamVestingTimeAtSupply[_time] == 0
534    @post __post.totalTokenSupply == totalTokenSupply + teamVestingTimeAtSupply[
535          _time]
536    @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingTimeAtSupply[_time]
537 */

```

Line 537-556 in File vanta.sol

```

537 function teamIssueVesting(address _to, uint _time) onlyIssuer public
538 {
539     require(saleTime == false);
540     require(teamVestingTime >= _time);
541
542     uint time = now;
543     require( ( ( endSaleTime + (_time * teamVestingDate) ) < time ) && (
544         teamVestingTimeAtSupply[_time] > 0 ) );
545
546     uint tokens = teamVestingTimeAtSupply[_time];
547
548     require(maxTeamSupply >= tokenIssuedTeam.add(tokens));
549
550     balances[_to] = balances[_to].add(tokens);
551     teamVestingTimeAtSupply[_time] = 0;
552
553     totalTokenSupply = totalTokenSupply.add(tokens);
554     tokenIssuedTeam = tokenIssuedTeam.add(tokens);
555
556     emit TeamIssue(_to, tokens);
557 }


```

✓ The code meets the specification

Formal Verification Request 23

advisorIssueVesting

 12, May 2019

 6023.56 ms

Line 558-569 in File vanta.sol

```

558  /*@CTK advisorIssueVesting
559      @tag assume_completion
560      @post !saleTime
561      @post advisorVestingTime >= _time
562      @post ((endSaleTime + _time * advisorVestingDate) < now) &&
563          (advisorVestingTimeAtSupply[_time] > 0)
564      @post maxAdvisorSupply >= tokenIssuedAdvisor + advisorVestingTimeAtSupply[_time]
565      @post __post.balances[_to] == balances[_to] + advisorVestingTimeAtSupply[_time]
566      @post __post.advisorVestingTimeAtSupply[_time] == 0
567      @post __post.totalTokenSupply == totalTokenSupply + advisorVestingTimeAtSupply[
568          _time]
569      @post __post.tokenIssuedAdvisor == tokenIssuedAdvisor +
570          advisorVestingTimeAtSupply[_time]
571  */

```

Line 570-589 in File vanta.sol

```

570  function advisorIssueVesting(address _to, uint _time) onlyIssuer public
571  {
572      require(saleTime == false);
573      require(advisorVestingTime >= _time);
574
575      uint time = now;
576      require( ( ( endSaleTime + (_time * advisorVestingDate) ) < time ) && (
577          advisorVestingTimeAtSupply[_time] > 0 ) );
578
579      uint tokens = advisorVestingTimeAtSupply[_time];
580
581      require(maxAdvisorSupply >= tokenIssuedAdvisor.add(tokens));
582
583      balances[_to] = balances[_to].add(tokens);
584      advisorVestingTimeAtSupply[_time] = 0;
585
586      totalTokenSupply = totalTokenSupply.add(tokens);
587      tokenIssuedAdvisor = tokenIssuedAdvisor.add(tokens);
588
589      emit AdvisorIssue(_to, tokens);
590  }


```

 The code meets the specification

Formal Verification Request 24

isTransferable

 12, May 2019

 3.51 ms

Line 595-598 in File vanta.sol



```
595  /*@CTK isTransferable
596      @post !tokenLock || msg.sender == manager -> __return
597      @post !__return -> tokenLock && msg.sender != manager
598  */
```

Line 599-611 in File vanta.sol

```
599  function isTransferable() private view returns (bool)
600  {
601      if(tokenLock == false)
602      {
603          return true;
604      }
605      else if(msg.sender == manager)
606      {
607          return true;
608      }
609
610      return false;
611  }
```

✓ The code meets the specification

Formal Verification Request 25

setTokenUnlock

📅 12, May 2019

🕒 59.94 ms

Line 613-619 in File vanta.sol

```
613  /*@CTK setTokenUnlock
614      @tag assume_completion
615      @post msg.sender == manager
616      @post tokenLock
617      @post !saleTime
618      @post !__post.tokenLock
619  */
```

Line 620-626 in File vanta.sol

```
620  function setTokenUnlock() onlyManager public
621  {
622      require(tokenLock == true);
623      require(saleTime == false);
624
625      tokenLock = false;
626  }
```

✓ The code meets the specification

Formal Verification Request 26

setTokenLock

📅 12, May 2019

🕒 38.98 ms

Line 628-633 in File vanta.sol

```
628  /*@CTK setTokenLock
629      @tag assume_completion
630      @post msg.sender == manager
631      @post !tokenLock
632      @post __post.tokenLock
633  */
```

Line 634-639 in File vanta.sol

```
634  function setTokenLock() onlyManager public
635  {
636      require(tokenLock == false);
637
638      tokenLock = true;
639  }
```

✓ The code meets the specification

Formal Verification Request 27

burnToken

📅 12, May 2019

🕒 768.58 ms

Line 720-727 in File vanta.sol

```
720  /*@CTK burnToken
721      @tag assume_completion
722      @post msg.sender == manager
723      @post balances[msg.sender] >= _value * E18
724      @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
725      @post __post.burnTokenSupply == burnTokenSupply + _value * E18
726      @post __post.totalTokenSupply == totalTokenSupply - _value * E18
727  */
```

Line 728-740 in File vanta.sol

```
728  function burnToken(uint _value) onlyManager public
729  {
730      uint tokens = _value * E18;
731
732      require(balances[msg.sender] >= tokens);
733
734      balances[msg.sender] = balances[msg.sender].sub(tokens);
735
736      burnTokenSupply = burnTokenSupply.add(tokens);
737      totalTokenSupply = totalTokenSupply.sub(tokens);
738
739      emit Burn(msg.sender, tokens);
740  }
```

✓ The code meets the specification




Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File vanta.sol

```
1 pragma solidity ^0.5.1;
```

 Only these compiler versions are safe to compile your code: 0.5.1, 0.5.2, 0.5.3, 0.5.4, 0.5.6

TIMESTAMP_DEPENDENCY

Line 542 in File vanta.sol

```
542 uint time = now;
```

 "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 575 in File vanta.sol

```
575 uint time = now;
```

 "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 646 in File vanta.sol

```
646 uint time = now;
```

 "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 681 in File vanta.sol

```
681 uint time = now;
```

 "now" can be influenced by minors to some degree

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- `VANTA_Token.sol` `d23fbdeb796294a79fc1bfc377f520119ed37fd4854d72a76d3d20bccd0cba03'`

Summary

CertiK team is invited by The Vanta Network team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checking as well as manual reviews by smart contract experts. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and The Vanta Network team has been actively giving us updates for the source code and feedback about the business logics.

At this point the Vanta Network team didn't provide other repositories sources as testing and documentation reference. We recommend having more unit tests coverage together with documentation to simulate potential use cases and walk through the functionalities to token holders, especially those super admin privileges that may impact the decentralized nature. Meanwhile, we are glad to see that Vanta Network team takes transparency seriously (i.e. supply and issue mechanism for each party is strictly written and followed to against any potential mischievous behaviors) and implement the lockup schedule with great care. However on the other hand, we recommend to decouple such features into different components so as to have a much cleaner token contract without any add-ons that purely for the purpose of lockup.

Overall we found the `VANTA_Token.sol` contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrative controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is 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 seeking multiple opinions, more test coverage and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

`VANTA_Token.sol`

- `SafeMath.div(uint256a, uint256 b)` – it is recommended to check case when `b == 0` and quit with an error message.
- `balanceOf(address _who)` – Use `.add()` instead of `+` operator, though it is a view function and there is a restriction from another function that would prevent overflow.
- `transfer(address _to, uint _value)` – check `_to` is not `address(0x0)`.



- `transferFrom(address _from, address _to, uint _value)` – check `_to` is not `address(0x0)`.
- `privateIssue(address _to, uint _value)` – it is recommended to have more detailed comment explaining how the token got released for private sale investors.
- `endSale()` – `teamVestingTimeAtSupply` is a `mapping (uint => uint)`, thus the value is set to 0 per item. In the for loop, `teamVestingTimeAtSupply[i] = teamVestingTimeAtSupply[i].add(teamVestingSupplyPerTime)`; could also be written like `array[i] = vest_amount` for better readability.

Source Code with CertiK Labels

File vanta.sol

```

1  pragma solidity ^0.5.1;
2
3  // Made By Tom Jung
4
5  library SafeMath
6  {
7      /*@CTK SafeMath_mul
8       @tag spec
9       @post __reverted == __has_assertion_failure
10      @post __has_assertion_failure == __has_overflow
11      @post __reverted == false -> __return == a * b
12      @post msg == msg__post
13      @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
14      @post __addr_map == __addr_map__post
15      */
16      function mul(uint256 a, uint256 b) internal pure returns (uint256)
17      {
18          uint256 c = a * b;
19          assert(a == 0 || c / a == b);
20
21          return c;
22      }
23
24      /*@CTK SafeMath_div
25       @tag spec
26       @pre b != 0
27       @post __reverted == __has_assertion_failure
28       @post __has_overflow == true -> __has_assertion_failure == true
29       @post __reverted == false -> __return == a / b
30       @post msg == msg__post
31       @post (b == 0) == __has_assertion_failure
32       @post __addr_map == __addr_map__post
33      */
34      function div(uint256 a, uint256 b) internal pure returns (uint256)
35      {
36          uint256 c = a / b;
37
38          return c;
39      }
40
41      /*@CTK SafeMath_sub
42       @tag spec
43       @post __reverted == __has_assertion_failure
44       @post __has_overflow == true -> __has_assertion_failure == true
45       @post __reverted == false -> __return == a - b
46       @post msg == msg__post
47       @post (b > a) == __has_assertion_failure
48       @post __addr_map == __addr_map__post
49      */
50      function sub(uint256 a, uint256 b) internal pure returns (uint256)
51      {
52          assert(b <= a);
53
54          return a - b;

```

```

55     }
56
57     /*@CTK SafeMath_add
58     @tag spec
59     @post __reverted == __has_assertion_failure
60     @post __has_assertion_failure == __has_overflow
61     @post __reverted == false -> __return == a + b
62     @post msg == msg__post
63     @post (a + b < a) == __has_assertion_failure
64     @post __addr_map == __addr_map__post
65     */
66     function add(uint256 a, uint256 b) internal pure returns (uint256)
67     {
68         uint256 c = a + b;
69         assert(c >= a);
70
71         return c;
72     }
73 }
74
75 contract OwnerHelper
76 {
77     address public master;
78     address public issuer;
79     address public manager;
80
81     event ChangeMaster(address indexed _from, address indexed _to);
82     event ChangeIssuer(address indexed _from, address indexed _to);
83     event ChangeManager(address indexed _from, address indexed _to);
84
85     modifier onlyMaster
86     {
87         require(msg.sender == master);
88         _;
89     }
90
91     modifier onlyIssuer
92     {
93         require(msg.sender == issuer);
94         _;
95     }
96
97     modifier onlyManager
98     {
99         require(msg.sender == manager);
100         _;
101     }
102
103     /*@CTK OwnerHelper
104     @post __post.master == msg.sender
105     */
106     constructor() public
107     {
108         master = msg.sender;
109     }
110
111     /*@CTK transferMastership
112     @tag assume_completion

```

```

113     @post msg.sender == master
114     @post _to != master
115     @post _to != issuer
116     @post _to != manager
117     @post _to != address(0x0)
118     @post __post.master == _to
119     */
120 function transferMastership(address _to) onlyMaster public
121 {
122     require(_to != master);
123     require(_to != issuer);
124     require(_to != manager);
125     require(_to != address(0x0));
126
127     address from = master;
128     master = _to;
129
130     emit ChangeMaster(from, _to);
131 }
132
133 /*@CTK transferIssuer
134   @tag assume_completion
135   @post msg.sender == master
136   @post _to != master
137   @post _to != issuer
138   @post _to != manager
139   @post _to != address(0x0)
140   @post __post.issuer == _to
141   */
142 function transferIssuer(address _to) onlyMaster public
143 {
144     require(_to != master);
145     require(_to != issuer);
146     require(_to != manager);
147     require(_to != address(0x0));
148
149     address from = issuer;
150     issuer = _to;
151
152     emit ChangeIssuer(from, _to);
153 }
154
155 /*@CTK transferManager
156   @tag assume_completion
157   @post msg.sender == master
158   @post _to != master
159   @post _to != issuer
160   @post _to != manager
161   @post _to != address(0x0)
162   @post __post.manager == _to
163   */
164 function transferManager(address _to) onlyMaster public
165 {
166     require(_to != master);
167     require(_to != issuer);
168     require(_to != manager);
169     require(_to != address(0x0));
170

```

```

171     address from = manager;
172     manager = _to;
173
174     emit ChangeManager(from, _to);
175 }
176 }
177
178 contract ERC20Interface
179 {
180     event Transfer( address indexed _from, address indexed _to, uint _value);
181     event Approval( address indexed _owner, address indexed _spender, uint _value);
182
183     function totalSupply() view public returns (uint _supply);
184     function balanceOf( address _who ) public view returns (uint _value);
185     function transfer( address _to, uint _value) public returns (bool _success);
186     function approve( address _spender, uint _value ) public returns (bool _success);
187     function allowance( address _owner, address _spender ) public view returns (uint
        _allowance);
188     function transferFrom( address _from, address _to, uint _value) public returns (
        bool _success);
189 }
190
191 contract VantaToken is ERC20Interface, OwnerHelper
192 {
193     using SafeMath for uint;
194
195     string public name;
196     uint public decimals;
197     string public symbol;
198
199     uint constant private E18 = 1000000000000000000;
200     uint constant private month = 2592000;
201
202     uint constant public maxTotalSupply = 56200000000 * E18;
203
204     uint constant public maxSaleSupply = 19670000000 * E18;
205     uint constant public maxBdevSupply = 8430000000 * E18;
206     uint constant public maxMktSupply = 8430000000 * E18;
207     uint constant public maxRndSupply = 8430000000 * E18;
208     uint constant public maxTeamSupply = 5620000000 * E18;
209     uint constant public maxReserveSupply = 2810000000 * E18;
210     uint constant public maxAdvisorSupply = 2810000000 * E18;
211
212     uint constant public teamVestingSupplyPerTime = 351250000 * E18;
213     uint constant public advisorVestingSupplyPerTime = 702500000 * E18;
214     uint constant public teamVestingDate = 2 * month;
215     uint constant public teamVestingTime = 16;
216     uint constant public advisorVestingDate = 3 * month;
217     uint constant public advisorVestingTime = 4;
218
219     uint public totalTokenSupply;
220
221     uint public tokenIssuedSale;
222     uint public privateIssuedSale;
223     uint public publicIssuedSale;
224     uint public tokenIssuedBdev;
225     uint public tokenIssuedMkt;
226     uint public tokenIssuedRnd;

```

```

227     uint public tokenIssuedTeam;
228     uint public tokenIssuedReserve;
229     uint public tokenIssuedAdvisor;
230
231     uint public burnTokenSupply;
232
233     mapping (address => uint) public balances;
234     mapping (address => mapping ( address => uint )) public approvals;
235
236     mapping (address => uint) public privateFirstWallet;
237
238     mapping (address => uint) public privateSecondWallet;
239
240     mapping (uint => uint) public teamVestingTimeAtSupply;
241     mapping (uint => uint) public advisorVestingTimeAtSupply;
242
243     bool public tokenLock = true;
244     bool public saleTime = true;
245     uint public endSaleTime = 0;
246
247     event Burn(address indexed _from, uint _value);
248
249     event SaleIssue(address indexed _to, uint _tokens);
250     event BdevIssue(address indexed _to, uint _tokens);
251     event MktIssue(address indexed _to, uint _tokens);
252     event RndIssue(address indexed _to, uint _tokens);
253     event TeamIssue(address indexed _to, uint _tokens);
254     event ReserveIssue(address indexed _to, uint _tokens);
255     event AdvisorIssue(address indexed _to, uint _tokens);
256
257     event TokenUnLock(address indexed _to, uint _tokens);
258
259     /*@CTK VantaToken
260     @tag assume_completion
261     @post __post.name == "VANTA Token"
262     @post __post.decimals == 18
263     @post __post.symbol == "VNT"
264     @post __post.totalTokenSupply == 0
265     @post __post.tokenIssuedSale == 0
266     @post __post.tokenIssuedBdev == 0
267     @post __post.tokenIssuedMkt == 0
268     @post __post.tokenIssuedRnd == 0
269     @post __post.tokenIssuedTeam == 0
270     @post __post.tokenIssuedReserve == 0
271     @post __post.tokenIssuedAdvisor == 0
272     @post maxTotalSupply == maxSaleSupply + maxBdevSupply + maxMktSupply +
        maxRndSupply + maxTeamSupply + maxReserveSupply + maxAdvisorSupply
273     @post maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime
274     @post maxAdvisorSupply == advisorVestingSupplyPerTime * advisorVestingTime
275     */
276     constructor() public
277     {
278         name      = "VANTA Token";
279         decimals   = 18;
280         symbol     = "VNT";
281
282         totalTokenSupply = 0;
283

```



```

284     tokenIssuedSale    = 0;
285     tokenIssuedBdev    = 0;
286     tokenIssuedMkt     = 0;
287     tokenIssuedRnd     = 0;
288     tokenIssuedTeam    = 0;
289     tokenIssuedReserve = 0;
290     tokenIssuedAdvisor = 0;
291
292     require(maxTotalSupply == maxSaleSupply + maxBdevSupply + maxMktSupply +
293             maxRndSupply + maxTeamSupply + maxReserveSupply + maxAdvisorSupply);
294     // CTK: No safemath?
295     require(maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime);
296     require(maxAdvisorSupply == advisorVestingSupplyPerTime * advisorVestingTime);
297 }
298 // ERC - 20 Interface -----
299
300 /*@CTK totalSupply
301    @post __return == totalTokenSupply
302 */
303 function totalSupply() view public returns (uint)
304 {
305     return totalTokenSupply;
306 }
307
308 /*@CTK balanceOf
309    @tag assume_completion
310    @post __return == balances[_who] + privateFirstWallet[_who] +
311            privateSecondWallet[_who]
312 */
313 function balanceOf(address _who) view public returns (uint)
314 {
315     uint balance = balances[_who];
316     balance = balance.add(privateFirstWallet[_who] + privateSecondWallet[_who]);
317
318     return balance;
319 }
320
321 /*@CTK transfer
322    @tag assume_completion
323    @pre msg.sender != _to
324    @post balances[msg.sender] >= _value
325    @post __post.balances[msg.sender] == balances[msg.sender] - _value
326    @post __post.balances[_to] == balances[_to] + _value
327 */
328 function transfer(address _to, uint _value) public returns (bool)
329 {
330     require(isTransferable() == true);
331     require(balances[msg.sender] >= _value);
332
333     balances[msg.sender] = balances[msg.sender].sub(_value);
334     balances[_to] = balances[_to].add(_value);
335
336     emit Transfer(msg.sender, _to, _value);
337
338     return true;
339 }
340

```

```

341  /*@CTK approve
342      @tag assume_completion
343      @pre msg.sender != _spender
344      @post balances[msg.sender] >= _value
345      @post __post.approvals[msg.sender][_spender] == _value
346  */
347  function approve(address _spender, uint _value) public returns (bool)
348  {
349      require(isTransferable() == true);
350      require(balances[msg.sender] >= _value);
351
352      approvals[msg.sender][_spender] = _value;
353
354      emit Approval(msg.sender, _spender, _value);
355
356      return true;
357  }
358
359  /*@CTK allowance
360      @post __return == approvals[_owner][_spender]
361  */
362  function allowance(address _owner, address _spender) view public returns (uint)
363  {
364      return approvals[_owner][_spender];
365  }
366
367  /*@CTK transferFrom
368      @tag assume_completion
369      @pre _from != _to
370      @post balances[_from] >= _value
371      @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
          _value
372      @post __post.balances[_from] == balances[_from] - _value
373      @post __post.balances[_to] == balances[_to] + _value
374  */
375  function transferFrom(address _from, address _to, uint _value) public returns (
      bool)
376  {
377      require(isTransferable() == true);
378      require(balances[_from] >= _value);
379      require(approvals[_from][msg.sender] >= _value);
380
381      approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
382      balances[_from] = balances[_from].sub(_value);
383      balances[_to] = balances[_to].add(_value);
384
385      emit Transfer(_from, _to, _value);
386
387      return true;
388  }
389
390  // -----
391
392  // Issue Function -----
393  /*@CTK privateIssue
394      @tag assume_completion
395      @post maxSaleSupply >= tokenIssuedSale + _value * E18
396      @post __post.balances[_to] == balances[_to] + _value * E18 * 435 / 1000

```

```

397     @post __post.privateFirstWallet[_to] == privateFirstWallet[_to] + _value * E18 *
        435 / 1000
398     @post __post.privateSecondWallet[_to] == privateSecondWallet[_to] + _value * E18
        * 435 / 1000
399     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
400     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
401     @post __post.privateIssuedSale == privateIssuedSale + _value * E18
402     */
403     function privateIssue(address _to, uint _value) onlyIssuer public
404     {
405         uint tokens = _value * E18;
406         require(maxSaleSupply >= tokenIssuedSale.add(tokens));
407
408         balances[_to] = balances[_to].add( tokens.mul(435)/1000 );
409         privateFirstWallet[_to] = privateFirstWallet[_to].add( tokens.mul(435)
            /1000 );
410         privateSecondWallet[_to] = privateSecondWallet[_to].add( tokens.mul(130)
            /1000 );
411
412         totalTokenSupply = totalTokenSupply.add(tokens);
413         tokenIssuedSale = tokenIssuedSale.add(tokens);
414         privateIssuedSale = privateIssuedSale.add(tokens);
415
416         emit SaleIssue(_to, tokens);
417     }
418
419     /*@CTK publicIssue
420     @tag assume_completion
421     @post maxSaleSupply >= tokenIssuedSale + _value * E18
422     @post __post.balances[_to] == balances[_to] + _value * E18
423     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
424     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
425     @post __post.publicIssuedSale == publicIssuedSale + _value * E18
426     */
427     function publicIssue(address _to, uint _value) onlyIssuer public
428     {
429         uint tokens = _value * E18;
430         require(maxSaleSupply >= tokenIssuedSale.add(tokens));
431
432         balances[_to] = balances[_to].add(tokens);
433
434         totalTokenSupply = totalTokenSupply.add(tokens);
435         tokenIssuedSale = tokenIssuedSale.add(tokens);
436         publicIssuedSale = publicIssuedSale.add(tokens);
437
438         emit SaleIssue(_to, tokens);
439     }
440
441     /*@CTK bdevIssue
442     @tag assume_completion
443     @post maxBdevSupply >= tokenIssuedBdev + _value * E18
444     @post __post.balances[_to] == balances[_to] + _value * E18
445     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
446     @post __post.tokenIssuedBdev == tokenIssuedBdev + _value * E18
447     */
448     function bdevIssue(address _to, uint _value) onlyIssuer public
449     {
450         uint tokens = _value * E18;

```

```

451     require(maxBdevSupply >= tokenIssuedBdev.add(tokens));
452
453     balances[_to] = balances[_to].add(tokens);
454
455     totalTokenSupply = totalTokenSupply.add(tokens);
456     tokenIssuedBdev = tokenIssuedBdev.add(tokens);
457
458     emit BdevIssue(_to, tokens);
459 }
460
461 /*@CTK mktIssue
462   @tag assume_completion
463   @post maxMktSupply >= tokenIssuedMkt + _value * E18
464   @post __post.balances[_to] == balances[_to] + _value * E18
465   @post __post.totalTokenSupply == totalTokenSupply + _value * E18
466   @post __post.tokenIssuedMkt == tokenIssuedMkt + _value * E18
467 */
468 function mktIssue(address _to, uint _value) onlyIssuer public
469 {
470     uint tokens = _value * E18;
471     require(maxMktSupply >= tokenIssuedMkt.add(tokens));
472
473     balances[_to] = balances[_to].add(tokens);
474
475     totalTokenSupply = totalTokenSupply.add(tokens);
476     tokenIssuedMkt = tokenIssuedMkt.add(tokens);
477
478     emit MktIssue(_to, tokens);
479 }
480
481 /*@CTK rndIssue
482   @tag assume_completion
483   @post maxRndSupply >= tokenIssuedRnd + _value * E18
484   @post __post.balances[_to] == balances[_to] + _value * E18
485   @post __post.totalTokenSupply == totalTokenSupply + _value * E18
486   @post __post.tokenIssuedRnd == tokenIssuedRnd + _value * E18
487 */
488 function rndIssue(address _to, uint _value) onlyIssuer public
489 {
490     uint tokens = _value * E18;
491     require(maxRndSupply >= tokenIssuedRnd.add(tokens));
492
493     balances[_to] = balances[_to].add(tokens);
494
495     totalTokenSupply = totalTokenSupply.add(tokens);
496     tokenIssuedRnd = tokenIssuedRnd.add(tokens);
497
498     emit RndIssue(_to, tokens);
499 }
500
501 /*@CTK reserveIssue
502   @tag assume_completion
503   @post maxReserveSupply >= tokenIssuedReserve + _value * E18
504   @post __post.balances[_to] == balances[_to] + _value * E18
505   @post __post.totalTokenSupply == totalTokenSupply + _value * E18
506   @post __post.tokenIssuedReserve == tokenIssuedReserve + _value * E18
507 */
508 function reserveIssue(address _to, uint _value) onlyIssuer public

```

```

509 {
510     uint tokens = _value * E18;
511     require(maxReserveSupply >= tokenIssuedReserve.add(tokens));
512
513     balances[_to] = balances[_to].add(tokens);
514
515     totalTokenSupply = totalTokenSupply.add(tokens);
516     tokenIssuedReserve = tokenIssuedReserve.add(tokens);
517
518     emit ReserveIssue(_to, tokens);
519 }
520
521 // ----
522
523 // Vesting Issue Function -----
524 /*@CTK teamIssueVesting
525    @tag assume_completion
526    @post msg.sender == issuer
527    @post !saleTime
528    @post teamVestingTime >= _time
529    @post (endSaleTime + _time * teamVestingDate < now) &&
530          (teamVestingTimeAtSupply[_time] > 0)
531    @post maxTeamSupply >= tokenIssuedTeam + teamVestingTimeAtSupply[_time]
532    @post __post.balances[_to] == balances[_to] + teamVestingTimeAtSupply[_time]
533    @post __post.teamVestingTimeAtSupply[_time] == 0
534    @post __post.totalTokenSupply == totalTokenSupply + teamVestingTimeAtSupply[
535          _time]
536    @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingTimeAtSupply[_time]
537 */
538 function teamIssueVesting(address _to, uint _time) onlyIssuer public
539 {
540     require(saleTime == false);
541     require(teamVestingTime >= _time);
542
543     uint time = now;
544     require( ( ( endSaleTime + (_time * teamVestingDate) ) < time ) && (
545         teamVestingTimeAtSupply[_time] > 0 ) );
546
547     uint tokens = teamVestingTimeAtSupply[_time];
548
549     require(maxTeamSupply >= tokenIssuedTeam.add(tokens));
550
551     balances[_to] = balances[_to].add(tokens);
552     teamVestingTimeAtSupply[_time] = 0;
553
554     totalTokenSupply = totalTokenSupply.add(tokens);
555     tokenIssuedTeam = tokenIssuedTeam.add(tokens);
556
557     emit TeamIssue(_to, tokens);
558 }
559
560 /*@CTK advisorIssueVesting
561    @tag assume_completion
562    @post !saleTime
563    @post advisorVestingTime >= _time
564    @post ((endSaleTime + _time * advisorVestingDate) < now) &&
565          (advisorVestingTimeAtSupply[_time] > 0)
566    @post maxAdvisorSupply >= tokenIssuedAdvisor + advisorVestingTimeAtSupply[_time]

```

```

565     @post __post.balances[_to] == balances[_to] + advisorVestingTimeAtSupply[_time]
566     @post __post.advisorVestingTimeAtSupply[_time] == 0
567     @post __post.totalTokenSupply == totalTokenSupply + advisorVestingTimeAtSupply[
        _time]
568     @post __post.tokenIssuedAdvisor == tokenIssuedAdvisor +
        advisorVestingTimeAtSupply[_time]
569     */
570     function advisorIssueVesting(address _to, uint _time) onlyIssuer public
571     {
572         require(saleTime == false);
573         require(advisorVestingTime >= _time);
574
575         uint time = now;
576         require( ( ( endSaleTime + (_time * advisorVestingDate) ) < time ) && (
            advisorVestingTimeAtSupply[_time] > 0 ) );
577
578         uint tokens = advisorVestingTimeAtSupply[_time];
579
580         require(maxAdvisorSupply >= tokenIssuedAdvisor.add(tokens));
581
582         balances[_to] = balances[_to].add(tokens);
583         advisorVestingTimeAtSupply[_time] = 0;
584
585         totalTokenSupply = totalTokenSupply.add(tokens);
586         tokenIssuedAdvisor = tokenIssuedAdvisor.add(tokens);
587
588         emit AdvisorIssue(_to, tokens);
589     }
590
591     // -----
592
593     // Lock Function -----
594     // CTK: make this a modifier?
595     /*@CTK isTransferable
596     @post !tokenLock || msg.sender == manager -> __return
597     @post !__return -> tokenLock && msg.sender != manager
598     */
599     function isTransferable() private view returns (bool)
600     {
601         if(tokenLock == false)
602         {
603             return true;
604         }
605         else if(msg.sender == manager)
606         {
607             return true;
608         }
609
610         return false;
611     }
612
613     /*@CTK setTokenUnlock
614     @tag assume_completion
615     @post msg.sender == manager
616     @post tokenLock
617     @post !saleTime
618     @post !__post.tokenLock
619     */

```

```

620 function setTokenUnlock() onlyManager public
621 {
622     require(tokenLock == true);
623     require(saleTime == false);
624
625     tokenLock = false;
626 }
627
628 /*@CTK setTokenLock
629    @tag assume_completion
630    @post msg.sender == manager
631    @post !tokenLock
632    @post __post.tokenLock
633 */
634 function setTokenLock() onlyManager public
635 {
636     require(tokenLock == false);
637
638     tokenLock = true;
639 }
640
641 function privateUnlock(address _to) onlyManager public
642 {
643     require(tokenLock == false);
644     require(saleTime == false);
645
646     uint time = now;
647     uint unlockTokens = 0;
648
649     if( (time >= endSaleTime.add(month)) && (privateFirstWallet[_to] > 0) )
650     {
651         balances[_to] = balances[_to].add(privateFirstWallet[_to]);
652         unlockTokens = unlockTokens.add(privateFirstWallet[_to]);
653         privateFirstWallet[_to] = 0;
654     }
655
656     if( (time >= endSaleTime.add(month * 2)) && (privateSecondWallet[_to] > 0) )
657     {
658         balances[_to] = balances[_to].add(privateSecondWallet[_to]);
659         unlockTokens = unlockTokens.add(privateSecondWallet[_to]);
660         privateSecondWallet[_to] = 0;
661     }
662
663     emit TokenUnLock(_to, unlockTokens);
664 }
665
666 // -----
667
668 // ETC / Burn Function -----
669
670 function () payable external
671 {
672     revert();
673 }
674
675 function endSale() onlyManager public
676 {
677     require(saleTime == true);

```



```

678
679     saleTime = false;
680
681     uint time = now;
682
683     endSaleTime = time;
684
685     /*CTK endSale_forloop
686     @inv forall j: uint. (j >= 0 /\ j < i) -> this.teamVestingTimeAtSupply[j] ==
687         this__pre.teamVestingTimeAtSupply[j] + teamVestingSupplyPerTime
688     @inv i <= teamVestingTime
689     @post i > teamVestingTime
690     @post !__should_return
691     */
692     for(uint i = 1; i <= teamVestingTime; i++)
693     {
694         teamVestingTimeAtSupply[i] = teamVestingTimeAtSupply[i].add(
695             teamVestingSupplyPerTime);
696     }
697
698     for(uint i = 1; i <= advisorVestingTime; i++)
699     {
700         advisorVestingTimeAtSupply[i] = advisorVestingTimeAtSupply[i].add(
701             advisorVestingSupplyPerTime);
702     }
703
704     function withdrawTokens(address _contract, uint _decimals, uint _value)
705         onlyManager public
706     {
707         if(_contract == address(0x0))
708         {
709             uint eth = _value.mul(10 ** _decimals);
710             msg.sender.transfer(eth);
711         }
712         else
713         {
714             uint tokens = _value.mul(10 ** _decimals);
715             ERC20Interface(_contract).transfer(msg.sender, tokens);
716
717             emit Transfer(address(0x0), msg.sender, tokens);
718         }
719     }
720
721     /*CTK burnToken
722     @tag assume_completion
723     @post msg.sender == manager
724     @post balances[msg.sender] >= _value * E18
725     @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
726     @post __post.burnTokenSupply == burnTokenSupply + _value * E18
727     @post __post.totalTokenSupply == totalTokenSupply - _value * E18
728     */
729     function burnToken(uint _value) onlyManager public
730     {
731         uint tokens = _value * E18;
732
733         require(balances[msg.sender] >= tokens);

```




```
733
734     balances[msg.sender] = balances[msg.sender].sub(tokens);
735
736     burnTokenSupply = burnTokenSupply.add(tokens);
737     totalTokenSupply = totalTokenSupply.sub(tokens);
738
739     emit Burn(msg.sender, tokens);
740 }
741
742 function close() onlyMaster public
743 {
744     selfdestruct(msg.sender);
745 }
746
747 // -----
748 }
```