CertiK Audit Report For Vocean

Request Date: 2019-05-08
Revision Date: 2019-05-10
Platform Name: Ethereum

Contents

Disclaimer

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

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by Vocean. This audit was conducted to discover issues and vulnerabilities in the source code of Vocean'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

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

- **VANToken.sol** `fc155dd5c4249609f83139a7a232119e500cf56a7b7252ed466e58a4a65965b9`

Etherscan reference: `0x95ca28bd2df36d4073b77980ac07ef316cf045bd`

Summary

CertiK team is invited by The Vocean 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 PlasmaPay team has been actively giving us updates for the source code and feedback about the business logics.

The VANToken source code has been deployed to ethereum mainnet at address `0x95ca28bd2df36d4073b77980ac07ef316cf045bd` by Thursday, March 28, 2019. It compiled with solidity compiler version ***v0.4.24+commit.e67f0147***. The `VANToken.sol` is a standard ERC20 token along with some additional operations:

At this point the Vocean team didn't provide other repositories sources as testing and documentation reference. We recommend to have 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.

Overall we found the `VANToken` contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrative privilege 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.

Source Code with CertiK Labels

File VANToken.sol

```
1  /**
2   * Source Code first verified at https://etherscan.io on Thursday, March 28, 2019
3   (UTC) */
4
5  pragma solidity ^0.4.24;
6
7  // File: openzeppelin-solidity/contracts/math/SafeMath.sol
8
9  /**
10  * @title SafeMath
11  * @dev Math operations with safety checks that revert on error
12  */
13 library SafeMath {
14
15   /**
16   * @dev Multiplies two numbers, reverts on overflow.
17   */
18   /*@CTK "SafeMath mul"
19     @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
20     @post !__reverted -> __return == a * b
21     @post !__reverted == !__has_overflow
22     @post !(__has_buf_overflow)
23     @post !(__has_assertion_failure)
24    */
25   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
26     // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
27     // benefit is lost if 'b' is also tested.
28     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
29     if (a == 0) {
30       return 0;
31     }
32
33     uint256 c = a * b;
34     require(c / a == b);
35
36     return c;
37   }
38
39   /**
40   * @dev Integer division of two numbers truncating the quotient, reverts on division
       by zero.
41   */
42   /*@CTK "SafeMath div"
43     @post b != 0 -> !__reverted
44     @post !__reverted -> __return == a / b
45     @post !__reverted -> !__has_overflow
46     @post !(__has_buf_overflow)
47     @post !(__has_assertion_failure)
48   */
49   function div(uint256 a, uint256 b) internal pure returns (uint256) {
50     require(b > 0); // Solidity only automatically asserts when dividing by 0
51     uint256 c = a / b;
52     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
53
```

```
54      return c;
55    }
56
57    /**
58    * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater
          than minuend).
59    */
60    /*@CTK "SafeMath sub"
61      @post (a < b) == __reverted
62      @post !__reverted -> __return == a - b
63      @post !__reverted -> !__has_overflow
64      @post !(__has_buf_overflow)
65      @post !(__has_assertion_failure)
66    */
67    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
68      require(b <= a);
69      uint256 c = a - b;
70
71      return c;
72    }
73
74    /**
75    * @dev Adds two numbers, reverts on overflow.
76    */
77    /*@CTK "SafeMath add"
78      @post (a + b < a || a + b < b) == __reverted
79      @post !__reverted -> __return == a + b
80      @post !__reverted -> !__has_overflow
81      @post !(__has_buf_overflow)
82      @post !(__has_assertion_failure)
83    */
84    function add(uint256 a, uint256 b) internal pure returns (uint256) {
85      uint256 c = a + b;
86      require(c >= a);
87
88      return c;
89    }
90
91    /**
92    * @dev Divides two numbers and returns the remainder (unsigned integer modulo),
93    * reverts when dividing by zero.
94    */
95    /*@CTK "SafeMath div"
96      @post b != 0 -> !__reverted
97      @post !__reverted -> __return == a % b
98      @post !__reverted -> !__has_overflow
99      @post !(__has_buf_overflow)
100     @post !(__has_assertion_failure)
101   */
102   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
103     require(b != 0);
104     return a % b;
105   }
106 }
107
108 // File: openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
109
110 /**
```

```
111  * @title ERC20 interface
112  * @dev see https://github.com/ethereum/EIPs/issues/20
113  */
114 interface IERC20 {
115   function totalSupply() external view returns (uint256);
116
117   function balanceOf(address who) external view returns (uint256);
118
119   function allowance(address owner, address spender)
120     external view returns (uint256);
121
122   function transfer(address to, uint256 value) external returns (bool);
123
124   function approve(address spender, uint256 value)
125     external returns (bool);
126
127   function transferFrom(address from, address to, uint256 value)
128     external returns (bool);
129
130   event Transfer(
131     address indexed from,
132     address indexed to,
133     uint256 value
134   );
135
136   event Approval(
137     address indexed owner,
138     address indexed spender,
139     uint256 value
140   );
141 }
142
143 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20.sol
144
145 /**
146  * @title Standard ERC20 token
147  *
148  * @dev Implementation of the basic standard token.
149  * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
150  * Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/
       master/smart_contract/FirstBloodToken.sol
151  */
152 contract ERC20 is IERC20 {
153   using SafeMath for uint256;
154
155   mapping (address => uint256) private _balances;
156
157   mapping (address => mapping (address => uint256)) private _allowed;
158
159   uint256 private _totalSupply;
160
161   /**
162   * @dev Total number of tokens in existence
163   */
164   //@CTK NO_OVERFLOW
165   //@CTK NO_BUF_OVERFLOW
166   //@CTK NO_ASF
167   /*@CTK "totalSupply correctness"
```

```
168     @post __return == _totalSupply
169    */
170   function totalSupply() public view returns (uint256) {
171     return _totalSupply;
172   }
173
174   /**
175   * @dev Gets the balance of the specified address.
176   * @param owner The address to query the balance of.
177   * @return An uint256 representing the amount owned by the passed address.
178   */
179   //@CTK NO_OVERFLOW
180   //@CTK NO_BUF_OVERFLOW
181   //@CTK NO_ASF
182   /*@CTK "balanceOf correctness"
183     @post __return == _balances[owner]
184    */
185   function balanceOf(address owner) public view returns (uint256) {
186     return _balances[owner];
187   }
188
189   /**
190    * @dev Function to check the amount of tokens that an owner allowed to a spender.
191    * @param owner address The address which owns the funds.
192    * @param spender address The address which will spend the funds.
193    * @return A uint256 specifying the amount of tokens still available for the spender
         .
194    */
195   //@CTK NO_OVERFLOW
196   //@CTK NO_BUF_OVERFLOW
197   //@CTK NO_ASF
198   /*@CTK "allowance correctness"
199     @post __return == _allowed[owner][spender]
200    */
201   function allowance(
202     address owner,
203     address spender
204    )
205     public
206     view
207     returns (uint256)
208   {
209     return _allowed[owner][spender];
210   }
211
212   /**
213   * @dev Transfer token for a specified address
214   * @param to The address to transfer to.
215   * @param value The amount to be transferred.
216   */
217   //@CTK NO_OVERFLOW
218   //@CTK NO_BUF_OVERFLOW
219   //@CTK NO_ASF
220   /*@CTK "transfer correctness"
221     @tag assume_completion
222     @post to != 0x0
223     @post value <= _balances[msg.sender]
```

```
224     @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
            value
225     @post to != msg.sender -> __post._balances[to] == _balances[to] + value
226     @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
227    */
228   function transfer(address to, uint256 value) public returns (bool) {
229     _transfer(msg.sender, to, value);
230     return true;
231   }
232
233   /**
234    * @dev Approve the passed address to spend the specified amount of tokens on behalf
           of msg.sender.
235    * Beware that changing an allowance with this method brings the risk that someone
           may use both the old
236    * and the new allowance by unfortunate transaction ordering. One possible solution
           to mitigate this
237    * race condition is to first reduce the spender's allowance to 0 and set the
           desired value afterwards:
238    * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
239    * @param spender The address which will spend the funds.
240    * @param value The amount of tokens to be spent.
241    */
242   //@CTK NO_OVERFLOW
243   //@CTK NO_BUF_OVERFLOW
244   //@CTK NO_ASF
245   /*@CTK "approve correctness"
246     @post spender == 0x0 -> __reverted
247     @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
248   */
249   function approve(address spender, uint256 value) public returns (bool) {
250     require(spender != address(0));
251
252     _allowed[msg.sender][spender] = value;
253     emit Approval(msg.sender, spender, value);
254     return true;
255   }
256
257   /**
258    * @dev Transfer tokens from one address to another
259    * @param from address The address which you want to send tokens from
260    * @param to address The address which you want to transfer to
261    * @param value uint256 the amount of tokens to be transferred
262    */
263   //@CTK NO_OVERFLOW
264   //@CTK NO_BUF_OVERFLOW
265   //@CTK NO_ASF
266   /*@CTK "transferFrom correctness"
267     @tag assume_completion
268     @post to != 0x0
269     @post value <= _balances[from] && value <= _allowed[from][msg.sender]
270     @post to != from -> __post._balances[from] == _balances[from] - value
271     @post to != from -> __post._balances[to] == _balances[to] + value
272     @post to == from -> __post._balances[from] == _balances[from]
273     @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
274    */
275   function transferFrom(
276     address from,
```

```
277     address to,
278     uint256 value
279   )
280     public
281     returns (bool)
282   {
283     require(value <= _allowed[from][msg.sender]);
284
285     _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
286     _transfer(from, to, value);
287     return true;
288   }
289
290   /**
291    * @dev Increase the amount of tokens that an owner allowed to a spender.
292    * approve should be called when allowed_[_spender] == 0. To increment
293    * allowed value is better to use this function to avoid 2 calls (and wait until
294    * the first transaction is mined)
295    * From MonolithDAO Token.sol
296    * @param spender The address which will spend the funds.
297    * @param addedValue The amount of tokens to increase the allowance by.
298    */
299   //@CTK NO_OVERFLOW
300   //@CTK NO_BUF_OVERFLOW
301   //@CTK NO_ASF
302   /*@CTK "increaseAllowance correctness"
303     @tag assume_completion
304     @post spender != 0x0
305     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
306   */
307   function increaseAllowance(
308     address spender,
309     uint256 addedValue
310   )
311     public
312     returns (bool)
313   {
314     require(spender != address(0));
315
316     _allowed[msg.sender][spender] = (
317       _allowed[msg.sender][spender].add(addedValue));
318     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
319     return true;
320   }
321
322   /**
323    * @dev Decrease the amount of tokens that an owner allowed to a spender.
324    * approve should be called when allowed_[_spender] == 0. To decrement
325    * allowed value is better to use this function to avoid 2 calls (and wait until
326    * the first transaction is mined)
327    * From MonolithDAO Token.sol
328    * @param spender The address which will spend the funds.
329    * @param subtractedValue The amount of tokens to decrease the allowance by.
330    */
331   //@CTK NO_OVERFLOW
332   //@CTK NO_BUF_OVERFLOW
333   //@CTK NO_ASF
```

```
334  /*@CTK "decreaseAllowance correctness"
335    @tag assume_completion
336    @post spender != 0x0
337    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
           subtractedValue
338  */
339  function decreaseAllowance(
340    address spender,
341    uint256 subtractedValue
342  )
343    public
344    returns (bool)
345  {
346    require(spender != address(0));
347
348    _allowed[msg.sender][spender] = (
349      _allowed[msg.sender][spender].sub(subtractedValue));
350    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
351    return true;
352  }
353
354  /**
355  * @dev Transfer token for a specified addresses
356  * @param from The address to transfer from.
357  * @param to The address to transfer to.
358  * @param value The amount to be transferred.
359  */
360  //@CTK NO_OVERFLOW
361  //@CTK NO_BUF_OVERFLOW
362  //@CTK NO_ASF
363  /*@CTK "_transfer correctness"
364    @tag assume_completion
365    @post to != 0x0
366    @post value <= _balances[from]
367    @post to != from -> __post._balances[from] == _balances[from] - value
368    @post to != from -> __post._balances[to] == _balances[to] + value
369    @post to == from -> __post._balances[from] == _balances[from]
370   */
371  function _transfer(address from, address to, uint256 value) internal {
372    require(value <= _balances[from]);
373    require(to != address(0));
374
375    _balances[from] = _balances[from].sub(value);
376    _balances[to] = _balances[to].add(value);
377    emit Transfer(from, to, value);
378  }
379
380  /**
381   * @dev Internal function that mints an amount of the token and assigns it to
382   * an account. This encapsulates the modification of balances such that the
383   * proper events are emitted.
384   * @param account The account that will receive the created tokens.
385   * @param value The amount that will be created.
386   */
387  //@CTK NO_OVERFLOW
388  //@CTK NO_BUF_OVERFLOW
389  //@CTK NO_ASF
390  /*@CTK "_mint correctness"
```

```
391     @tag assume_completion
392     @post account != 0x0
393     @post __post._balances[account] == _balances[account] + value
394     @post __post._totalSupply == _totalSupply + value
395    */
396   function _mint(address account, uint256 value) internal {
397     require(account != 0);
398     _totalSupply = _totalSupply.add(value);
399     _balances[account] = _balances[account].add(value);
400     emit Transfer(address(0), account, value);
401   }
402
403   /**
404    * @dev Internal function that burns an amount of the token of a given
405    * account.
406    * @param account The account whose tokens will be burnt.
407    * @param value The amount that will be burnt.
408    */
409   //@CTK NO_OVERFLOW
410   //@CTK NO_BUF_OVERFLOW
411   //@CTK NO_ASF
412   /*@CTK "_burn correctness"
413     @tag assume_completion
414     @post account != 0x0
415     @post value <= _balances[account]
416     @post __post._balances[account] == _balances[account] - value
417     @post __post._totalSupply == _totalSupply - value
418    */
419   function _burn(address account, uint256 value) internal {
420     require(account != 0);
421     require(value <= _balances[account]);
422
423     _totalSupply = _totalSupply.sub(value);
424     _balances[account] = _balances[account].sub(value);
425     emit Transfer(account, address(0), value);
426   }
427
428   /**
429    * @dev Internal function that burns an amount of the token of a given
430    * account, deducting from the sender's allowance for said account. Uses the
431    * internal burn function.
432    * @param account The account whose tokens will be burnt.
433    * @param value The amount that will be burnt.
434    */
435   //@CTK NO_OVERFLOW
436   //@CTK NO_BUF_OVERFLOW
437   //@CTK NO_ASF
438   /*@CTK "_burnFrom correctness"
439     @tag assume_completion
440     @post account != 0x0
441     @post value <= _balances[account] && value <= _allowed[account][msg.sender]
442     @post __post._balances[account] == _balances[account] - value
443     @post __post._totalSupply == _totalSupply - value
444     @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
          value
445    */
446   function _burnFrom(address account, uint256 value) internal {
447     require(value <= _allowed[account][msg.sender]);
```

```
448
449     // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
            ,
450     // this function needs to emit an event with the updated approval.
451     _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
452       value);
453     _burn(account, value);
454   }
455 }
456
457 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20Burnable.sol
458
459 /**
460  * @title Burnable Token
461  * @dev Token that can be irreversibly burned (destroyed).
462  */
463 contract ERC20Burnable is ERC20 {
464
465   /**
466    * @dev Burns a specific amount of tokens.
467    * @param value The amount of token to be burned.
468    */
469   //@CTK NO_OVERFLOW
470   //@CTK NO_BUF_OVERFLOW
471   //@CTK NO_ASF
472   function burn(uint256 value) public {
473     _burn(msg.sender, value);
474   }
475
476   /**
477    * @dev Burns a specific amount of tokens from the target address and decrements
           allowance
478    * @param from address The address which you want to send tokens from
479    * @param value uint256 The amount of token to be burned
480    */
481   //@CTK NO_OVERFLOW
482   //@CTK NO_BUF_OVERFLOW
483   //@CTK NO_ASF
484   function burnFrom(address from, uint256 value) public {
485     _burnFrom(from, value);
486   }
487 }
488
489 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol
490
491 /**
492  * @title ERC20Detailed token
493  * @dev The decimals are only for visualization purposes.
494  * All the operations are done using the smallest and indivisible token unit,
495  * just as on Ethereum all the operations are done in wei.
496  */
497 contract ERC20Detailed is IERC20 {
498   string private _name;
499   string private _symbol;
500   uint8 private _decimals;
501
502   //@CTK NO_OVERFLOW
503   //@CTK NO_BUF_OVERFLOW
```

```
504   //@CTK NO_ASF
505   /*@CTK "ERC20Detailed constructor correctness"
506     @post __post._name == name
507     @post __post._symbol == symbol
508     @post __post._decimals == decimals
509    */
510   constructor(string name, string symbol, uint8 decimals) public {
511     _name = name;
512     _symbol = symbol;
513     _decimals = decimals;
514   }
515
516   /**
517    * @return the name of the token.
518    */
519   //@CTK NO_OVERFLOW
520   //@CTK NO_BUF_OVERFLOW
521   //@CTK NO_ASF
522   /*@CTK "ERC20Detailed name correctness"
523     @post __return == _name
524    */
525   function name() public view returns(string) {
526     return _name;
527   }
528
529   /**
530    * @return the symbol of the token.
531    */
532   //@CTK NO_OVERFLOW
533   //@CTK NO_BUF_OVERFLOW
534   //@CTK NO_ASF
535   /*@CTK "ERC20Detailed symbol correctness"
536     @post __return == _symbol
537    */
538   function symbol() public view returns(string) {
539     return _symbol;
540   }
541
542   /**
543    * @return the number of decimals of the token.
544    */
545   //@CTK NO_OVERFLOW
546   //@CTK NO_BUF_OVERFLOW
547   //@CTK NO_ASF
548   /*@CTK "ERC20Detailed decimals correctness"
549     @post __return == _decimals
550    */
551   function decimals() public view returns(uint8) {
552     return _decimals;
553   }
554 }
555
556 // File: openzeppelin-solidity/contracts/access/Roles.sol
557
558 /**
559  * @title Roles
560  * @dev Library for managing addresses assigned to a Role.
561  */
```

```
562 library Roles {
563   struct Role {
564     mapping (address => bool) bearer;
565   }
566
567   /**
568    * @dev give an account access to this role
569    */
570   //@CTK NO_OVERFLOW
571   //@CTK NO_BUF_OVERFLOW
572   //@CTK NO_ASF
573   /*@CTK "Roles add correctness"
574     @post account == 0x0 -> __reverted
575     @post role.bearer[account] -> __reverted
576     @post account != 0x0 && !role.bearer[account] -> !__reverted
577    */
578   function add(Role storage role, address account) internal {
579     require(account != address(0));
580     require(!role.bearer[account]);
581
582     role.bearer[account] = true;
583   }
584
585   /**
586    * @dev remove an account's access to this role
587    */
588   //@CTK NO_OVERFLOW
589   //@CTK NO_BUF_OVERFLOW
590   //@CTK NO_ASF
591   /*@CTK "Roles add correctness"
592     @post account == 0x0 -> __reverted
593     @post !role.bearer[account] -> __reverted
594     @post account != 0x0 && role.bearer[account] -> !__reverted
595    */
596   function remove(Role storage role, address account) internal {
597     require(account != address(0));
598     require(role.bearer[account]);
599
600     role.bearer[account] = false;
601   }
602
603   /**
604    * @dev check if an account has this role
605    * @return bool
606    */
607   //@CTK NO_OVERFLOW
608   //@CTK NO_BUF_OVERFLOW
609   //@CTK NO_ASF
610   /*@CTK "Roles has correctness"
611     @post account == 0x0 -> __reverted
612     @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
613    */
614   function has(Role storage role, address account)
615     internal
616     view
617     returns (bool)
618   {
619     require(account != address(0));
```

```
620     return role.bearer[account];
621   }
622 }
623
624 // File: openzeppelin-solidity/contracts/access/roles/MinterRole.sol
625
626 contract MinterRole {
627   using Roles for Roles.Role;
628
629   event MinterAdded(address indexed account);
630   event MinterRemoved(address indexed account);
631
632   Roles.Role private minters;
633
634   //@CTK NO_OVERFLOW
635   //@CTK NO_BUF_OVERFLOW
636   //@CTK NO_ASF
637   /*@CTK "MinterRole constructor correctness"
638     @post msg.sender == 0x0 -> __reverted
639     @post minters.bearer[msg.sender] -> __reverted
640     @post msg.sender != 0x0 && !minters.bearer[msg.sender]
641           -> !__reverted && __post.minters.bearer[msg.sender]
642    */
643   constructor() internal {
644     _addMinter(msg.sender);
645   }
646
647   modifier onlyMinter() {
648     require(isMinter(msg.sender));
649     _;
650   }
651
652   //@CTK NO_OVERFLOW
653   //@CTK NO_BUF_OVERFLOW
654   //@CTK NO_ASF
655   /*@CTK "isMinter correctness"
656     @post account == 0x0 -> __reverted
657     @post account != 0x0 -> !__reverted && __return == minters.bearer[account]
658    */
659   function isMinter(address account) public view returns (bool) {
660     return minters.has(account);
661   }
662
663   //@CTK NO_OVERFLOW
664   //@CTK NO_BUF_OVERFLOW
665   //@CTK NO_ASF
666   /*@CTK "_addPauser correctness"
667     @post account == 0x0 -> __reverted
668     @post msg.sender == 0x0 -> __reverted
669     @post minters.bearer[account] -> __reverted
670     @post !minters.bearer[msg.sender] -> __reverted
671     @post account != 0x0 && !minters.bearer[account]
672           && msg.sender != 0x0 && minters.bearer[msg.sender]
673           -> !__reverted && __post.minters.bearer[account]
674    */
675   function addMinter(address account) public onlyMinter {
676     _addMinter(account);
677   }
```

```
678
679   //@CTK NO_OVERFLOW
680   //@CTK NO_BUF_OVERFLOW
681   //@CTK NO_ASF
682   /*@CTK "renouncePauser correctness"
683     @post msg.sender == 0x0 -> __reverted
684     @post !minters.bearer[msg.sender] -> __reverted
685     @post msg.sender != 0x0 && minters.bearer[msg.sender]
686           -> !__reverted && !__post.minters.bearer[msg.sender]
687    */
688   function renounceMinter() public {
689     _removeMinter(msg.sender);
690   }
691
692   //@CTK NO_OVERFLOW
693   //@CTK NO_BUF_OVERFLOW
694   //@CTK NO_ASF
695   /*@CTK "_addPauser correctness"
696     @post account == 0x0 -> __reverted
697     @post minters.bearer[account] -> __reverted
698     @post account != 0x0 && !minters.bearer[account]
699           -> !__reverted && __post.minters.bearer[account]
700    */
701   function _addMinter(address account) internal {
702     minters.add(account);
703     emit MinterAdded(account);
704   }
705
706   //@CTK NO_OVERFLOW
707   //@CTK NO_BUF_OVERFLOW
708   //@CTK NO_ASF
709   /*@CTK "_removePauser correctness"
710     @post account == 0x0 -> __reverted
711     @post !minters.bearer[account] -> __reverted
712     @post account != 0x0 && minters.bearer[account]
713           -> !__reverted && !__post.minters.bearer[account]
714    */
715   function _removeMinter(address account) internal {
716     minters.remove(account);
717     emit MinterRemoved(account);
718   }
719 }
720
721 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol
722
723 /**
724  * @title ERC20Mintable
725  * @dev ERC20 minting logic
726  */
727 contract ERC20Mintable is ERC20, MinterRole {
728   /**
729    * @dev Function to mint tokens
730    * @param to The address that will receive the minted tokens.
731    * @param value The amount of tokens to mint.
732    * @return A boolean that indicates if the operation was successful.
733    */
734   //@CTK NO_OVERFLOW
735   //@CTK NO_BUF_OVERFLOW
```

```
736   //@CTK NO_ASF
737   /*@CTK mint
738     @tag assume_completion
739     @post to != 0
740     @post __post._totalSupply == _totalSupply + value
741     @post __post._balances[to] == _balances[to] + value
742    */
743   function mint(
744     address to,
745     uint256 value
746   )
747     public
748     onlyMinter
749     returns (bool)
750   {
751     _mint(to, value);
752     return true;
753   }
754 }
755
756 // File: openzeppelin-solidity/contracts/access/roles/PauserRole.sol
757
758 contract PauserRole {
759   using Roles for Roles.Role;
760
761   event PauserAdded(address indexed account);
762   event PauserRemoved(address indexed account);
763
764   Roles.Role private pausers;
765
766   //@CTK NO_OVERFLOW
767   //@CTK NO_BUF_OVERFLOW
768   //@CTK NO_ASF
769   /*@CTK "PauserRole constructor correctness"
770     @post msg.sender == 0x0 -> __reverted
771     @post pausers.bearer[msg.sender] -> __reverted
772     @post msg.sender != 0x0 && !pausers.bearer[msg.sender]
773           -> !__reverted && __post.pausers.bearer[msg.sender]
774    */
775   constructor() internal {
776     _addPauser(msg.sender);
777   }
778
779   modifier onlyPauser() {
780     require(isPauser(msg.sender));
781     _;
782   }
783
784   //@CTK NO_OVERFLOW
785   //@CTK NO_BUF_OVERFLOW
786   //@CTK NO_ASF
787   /*@CTK "isBurner correctness"
788     @post account == 0x0 -> __reverted
789     @post account != 0x0 -> !__reverted && __return == pausers.bearer[account]
790    */
791   function isPauser(address account) public view returns (bool) {
792     return pausers.has(account);
793   }
```

```
794
795   //@CTK NO_OVERFLOW
796   //@CTK NO_BUF_OVERFLOW
797   //@CTK NO_ASF
798   /*@CTK "_addPauser correctness"
799     @post account == 0x0 -> __reverted
800     @post msg.sender == 0x0 -> __reverted
801     @post pausers.bearer[account] -> __reverted
802     @post !pausers.bearer[msg.sender] -> __reverted
803     @post account != 0x0 && !pausers.bearer[account]
804          && msg.sender != 0x0 && pausers.bearer[msg.sender]
805          -> !__reverted && __post.pausers.bearer[account]
806    */
807   function addPauser(address account) public onlyPauser {
808     _addPauser(account);
809   }
810
811   //@CTK NO_OVERFLOW
812   //@CTK NO_BUF_OVERFLOW
813   //@CTK NO_ASF
814   /*@CTK "renouncePauser correctness"
815     @post msg.sender == 0x0 -> __reverted
816     @post !pausers.bearer[msg.sender] -> __reverted
817     @post msg.sender != 0x0 && pausers.bearer[msg.sender]
818          -> !__reverted && !__post.pausers.bearer[msg.sender]
819    */
820   function renouncePauser() public {
821     _removePauser(msg.sender);
822   }
823
824   //@CTK NO_OVERFLOW
825   //@CTK NO_BUF_OVERFLOW
826   //@CTK NO_ASF
827   /*@CTK "_addPauser correctness"
828     @post account == 0x0 -> __reverted
829     @post pausers.bearer[account] -> __reverted
830     @post account != 0x0 && !pausers.bearer[account]
831          -> !__reverted && __post.pausers.bearer[account]
832    */
833   function _addPauser(address account) internal {
834     pausers.add(account);
835     emit PauserAdded(account);
836   }
837
838   //@CTK NO_OVERFLOW
839   //@CTK NO_BUF_OVERFLOW
840   //@CTK NO_ASF
841   /*@CTK "_removePauser correctness"
842     @post account == 0x0 -> __reverted
843     @post !pausers.bearer[account] -> __reverted
844     @post account != 0x0 && pausers.bearer[account]
845          -> !__reverted && !__post.pausers.bearer[account]
846    */
847   function _removePauser(address account) internal {
848     pausers.remove(account);
849     emit PauserRemoved(account);
850   }
851 }
```

```
852
853 // File: openzeppelin-solidity/contracts/lifecycle/Pausable.sol
854
855 /**
856  * @title Pausable
857  * @dev Base contract which allows children to implement an emergency stop mechanism.
858  */
859 contract Pausable is PauserRole {
860   event Paused(address account);
861   event Unpaused(address account);
862
863   bool private _paused;
864
865   //@CTK NO_OVERFLOW
866   //@CTK NO_BUF_OVERFLOW
867   //@CTK NO_ASF
868   /*@CTK "Pausable constructor correctness"
869     @post __post._paused == false
870    */
871   constructor() internal {
872     _paused = false;
873   }
874
875   /**
876    * @return true if the contract is paused, false otherwise.
877    */
878   //@CTK NO_OVERFLOW
879   //@CTK NO_BUF_OVERFLOW
880   //@CTK NO_ASF
881   /*@CTK "Pausable paused correctness"
882     @post __return == _paused
883    */
884   function paused() public view returns(bool) {
885     return _paused;
886   }
887
888   /**
889    * @dev Modifier to make a function callable only when the contract is not paused.
890    */
891   modifier whenNotPaused() {
892     require(!_paused);
893     _;
894   }
895
896   /**
897    * @dev Modifier to make a function callable only when the contract is paused.
898    */
899   modifier whenPaused() {
900     require(_paused);
901     _;
902   }
903
904   /**
905    * @dev called by the owner to pause, triggers stopped state
906    */
907   //@CTK NO_OVERFLOW
908   //@CTK NO_BUF_OVERFLOW
909   //@CTK NO_ASF
```

```
  /*@CTK "Pausable pause correctness"
    @post _paused -> __reverted
    @post msg.sender == 0x0 -> __reverted
    @post !pausers.bearer[msg.sender] -> __reverted
    @post !_paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
          -> __post._paused
   */
  function pause() public onlyPauser whenNotPaused {
    _paused = true;
    emit Paused(msg.sender);
  }

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "Pausable unpause correctness"
    @post !_paused -> __reverted
    @post msg.sender == 0x0 -> __reverted
    @post !pausers.bearer[msg.sender] -> __reverted
    @post _paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
          -> !__post._paused
   */
  function unpause() public onlyPauser whenPaused {
    _paused = false;
    emit Unpaused(msg.sender);
  }
}

// File: openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol

/**
 * @title Pausable token
 * @dev ERC20 modified with pausable transfers.
 **/
contract ERC20Pausable is ERC20, Pausable {

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "ERC20Pausable transfer correctness"
    @tag assume_completion
    @post to != 0x0
    @post value <= _balances[msg.sender]
    @post _paused == false
    @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
        value
    @post to != msg.sender -> __post._balances[to] == _balances[to] + value
    @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
   */
  function transfer(
    address to,
    uint256 value
  )
    public
    whenNotPaused
```

```
    returns (bool)
  {
    return super.transfer(to, value);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "ERC20Pausable transferFrom correctness"
    @tag assume_completion
    @post to != 0x0
    @post value <= _balances[from] && value <= _allowed[from][msg.sender]
    @post _paused == false
    @post to != from -> __post._balances[from] == _balances[from] - value
    @post to != from -> __post._balances[to] == _balances[to] + value
    @post to == from -> __post._balances[from] == _balances[from]
    @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
   */
  function transferFrom(
    address from,
    address to,
    uint256 value
  )
    public
    whenNotPaused
    returns (bool)
  {
    return super.transferFrom(from, to, value);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "ERC20Pausable approve correctness"
    @post spender == 0x0 -> __reverted
    @post _paused -> __reverted
    @post spender != 0x0 && !_paused
          -> __post._allowed[msg.sender][spender] == value
  */
  function approve(
    address spender,
    uint256 value
  )
    public
    whenNotPaused
    returns (bool)
  {
    return super.approve(spender, value);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "ERC20Pausable increaseAllowance correctness"
    @tag assume_completion
    @post spender != 0x0
    @post _paused == false
```

```
    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
        addedValue
  */
  function increaseAllowance(
    address spender,
    uint addedValue
  )
    public
    whenNotPaused
    returns (bool success)
  {
    return super.increaseAllowance(spender, addedValue);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "ERC20Pausable decreaseAllowance correctness"
    @tag assume_completion
    @post spender != 0x0
    @post _paused == false
    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
        subtractedValue
  */
  function decreaseAllowance(
    address spender,
    uint subtractedValue
  )
    public
    whenNotPaused
    returns (bool success)
  {
    return super.decreaseAllowance(spender, subtractedValue);
  }
}

// File: contracts/NETToken.sol

contract VANToken is ERC20Detailed, ERC20Pausable, ERC20Mintable, ERC20Burnable {

    /**
     * Initial supply is 1 billion tokens.
     */
    uint256 public constant INITIAL_SUPPLY = 1000000000 * (10 ** 18);

    /**
     * Constructor that gives msg.sender all of existing tokens.
     */
    constructor() public ERC20Detailed("Vocean Token", "VAN", 18) {
        _mint(msg.sender, INITIAL_SUPPLY);
    }

}
```