

CERTIK AUDIT REPORT FOR VALLIX



Request Date: 2019-08-27
Revision Date: 2019-08-30
Platform Name: Ethereum



Contents

Disclaimer	1
About CertiK	2
Exective Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Manual Review Notes	6
Static Analysis Results	8
Formal Verification Results	9
How to read	9
Source Code with CertiK Labels	33

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

About CertiK

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

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

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

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

Executive Summary

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

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

Vulnerability Classification

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

Critical

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

Medium

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

Low

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

Testing Summary

PASS

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

Aug 30, 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.	3	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.	3	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

- **ERC20Interface.sol**
42f714e48306fe818d71d8a763c5d7c703c7c582e4b2faa578fd0e56408b7730
- **OwnerHelper.sol**
df04d29467474905422ddb36ef75e92f632c120eb23698b4e0b0dd06a4066835
- **SafeMath.sol**
338d381fd57be4820b27bea197b7b23a05edd182ae7bf6910b8edb1c4a4f7036
- **VALLIXToken.sol**
43d26684308702ff29a409faed30e160496462711454ca7a6556dd20f6b4351e

Summary

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

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

VALLIXToken.sol

- `constructor()`: Recommend using `.add()` in `library SafeMath` instead of `+` though it is a constructor.
- `balanceOf(address _who)`: Recommend using `.add()` in `library SafeMath` instead of `+` 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)`.
- `transferFrom(address _from, address _to, uint _value)`: Check `_from` is not `_to`.
- `isTransferable()`: Recommend declaring as `modifier` to cover this functionality.
- `privateIssue(address _to, uint _value)`: Recommend having more detailed comment explaining how the token got released for private sale investors.
- `endSale()`: Since `teamVestingTimeAtSupply` is a `mapping (uint => uint)`, 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.

SafeMath.sol

- `div(uint256 a, uint256 b)`: Recommend using `require(b != 0)`; since `uint` is considered non-negative.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File VALLIXToken.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

TIMESTAMP_DEPENDENCY

Line 327 in File VALLIXToken.sol

```
327 uint time = now;
```

 "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 420 in File VALLIXToken.sol

```
420 uint time = now;
```

 "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 478 in File VALLIXToken.sol

```
478 uint time = now;
```

 "now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File OwnerHelper.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

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	<pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from][msg.sender] == 34 */ </pre>
Raw code location	Line 35-41 in File howtoread.sol
Raw code	<pre> 35 function transferFrom(address from, address to 36) { 37 balances[from] = balances[from].sub(tokens 38 allowed[from][msg.sender] = allowed[from][39 balances[to] = balances[to].add(tokens); 40 emit Transfer(from, to, tokens); 41 return true; </pre>
Counterexample	 This code violates the specification
Initial environment	<pre> 1 Counter Example: 2 Before Execution: 3 Input = { 4 from = 0x0 5 to = 0x0 6 tokens = 0x6c 7 } 8 This = 0 </pre>
Post environment	<pre> 52 } 53 balance: 0x0 54 } 55 } 56 57 After Execution: 58 Input = { 59 from = 0x0 60 to = 0x0 61 tokens = 0x6c </pre>

Formal Verification Request 1

totalSupply

📅 30, Aug 2019

🕒 6.92 ms

Line 107-109 in File VALLIXToken.sol

```
107  /*@CTK totalSupply
108      @post __return == totalTokenSupply
109  */
```

Line 110-113 in File VALLIXToken.sol

```
110  function totalSupply() view public returns (uint)
111  {
112      return totalTokenSupply;
113  }
```

✅ The code meets the specification.

Formal Verification Request 2

balanceOf

📅 30, Aug 2019

🕒 59.75 ms

Line 114-117 in File VALLIXToken.sol

```
114  /*@CTK balanceOf
115      @tag assume_completion
116      @post __return == (balances[_who] + privateFirstWallet[_who] +
117                      privateSecondWallet[_who] + privateThirdWallet[_who] + privateFourthWallet[
118                          _who] + privateFifthWallet[_who])
119  */
```

Line 118-124 in File VALLIXToken.sol

```
118  function balanceOf(address _who) view public returns (uint)
119  {
120      uint balance = balances[_who];
121      balance = balance.add(privateFirstWallet[_who] + privateSecondWallet[_who] +
122                          privateThirdWallet[_who] + privateFourthWallet[_who] + privateFifthWallet[
123                          _who]);
124      return balance;
125  }
```

✅ The code meets the specification.

Formal Verification Request 3

transfer

📅 30, Aug 2019

🕒 229.56 ms

Line 126-132 in File VALLIXToken.sol

```
126  /*@CTK transfer
127      @tag assume_completion
128      @pre msg.sender != _to
129      @post balances[msg.sender] >= _value
130      @post __post.balances[msg.sender] == balances[msg.sender] - _value
131      @post __post.balances[_to] == balances[_to] + _value
132  */
```

Line 133-144 in File VALLIXToken.sol

```
133  function transfer(address _to, uint _value) public returns (bool)
134  {
135      require(isTransferable() == true);
136      require(balances[msg.sender] >= _value);
137
138      balances[msg.sender] = balances[msg.sender].sub(_value);
139      balances[_to] = balances[_to].add(_value);
140
141      emit Transfer(msg.sender, _to, _value);
142
143      return true;
144  }
```

✓ The code meets the specification.

Formal Verification Request 4

approve



30, Aug 2019



68.33 ms

Line 146-151 in File VALLIXToken.sol

```
146  /*@CTK approve
147      @tag assume_completion
148      @pre msg.sender != _spender
149      @post balances[msg.sender] >= _value
150      @post __post.approvals[msg.sender][_spender] == _value
151  */
```

Line 152-162 in File VALLIXToken.sol

```
152  function approve(address _spender, uint _value) public returns (bool)
153  {
154      require(isTransferable() == true);
155      require(balances[msg.sender] >= _value);
156
157      approvals[msg.sender][_spender] = _value;
158
159      emit Approval(msg.sender, _spender, _value);
160
161      return true;
162  }
```

✓ The code meets the specification.

Formal Verification Request 5

allowance

📅 30, Aug 2019

🕒 6.6 ms

Line 164-166 in File VALLIXToken.sol

```
164  /*@CTK allowance
165      @post __return == approvals[_owner][_spender]
166  */
```

Line 167-170 in File VALLIXToken.sol

```
167  function allowance(address _owner, address _spender) view public returns (uint)
168  {
169      return approvals[_owner][_spender];
170  }
```

✅ The code meets the specification.

Formal Verification Request 6

transferFrom

📅 30, Aug 2019

🕒 440.79 ms

Line 172-179 in File VALLIXToken.sol

```
172  /*@CTK transferFrom
173      @tag assume_completion
174      @pre _from != _to
175      @post balances[_from] >= _value
176      @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
          _value
177      @post __post.balances[_from] == balances[_from] - _value
178      @post __post.balances[_to] == balances[_to] + _value
179  */
```

Line 180-193 in File VALLIXToken.sol

```
180  function transferFrom(address _from, address _to, uint _value) public returns (
181      bool)
182  {
183      require(isTransferable() == true);
184      require(balances[_from] >= _value);
185      require(approvals[_from][msg.sender] >= _value);
186      approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
187      balances[_from] = balances[_from].sub(_value);
188      balances[_to] = balances[_to].add(_value);
189
190      emit Transfer(_from, _to, _value);
191
192      return true;
193  }
```

✓ The code meets the specification.

Formal Verification Request 7

privateIssue

📅 30, Aug 2019

🕒 2146.97 ms

Line 194-207 in File VALLIXToken.sol

```

194  /*@CTK privateIssue
195     @tag assume_completion
196     @post msg.sender == issuer
197     @post maxSaleSupply >= tokenIssuedSale + (_value * E18)
198     @post __post.balances[_to] == balances[_to] + _value * E18 * 10 / 100
199     @post __post.privateFirstWallet[_to] == privateFirstWallet[_to] + _value * E18 *
      10 / 100
200     @post __post.privateSecondWallet[_to] == privateSecondWallet[_to] + _value * E18
      * 10 / 100
201     @post __post.privateThirdWallet[_to] == privateThirdWallet[_to] + _value * E18 *
      20 / 100
202     @post __post.privateFourthWallet[_to] == privateFourthWallet[_to] + _value * E18
      * 20 / 100
203     @post __post.privateFifthWallet[_to] == privateFifthWallet[_to] + _value * E18 *
      30 / 100
204     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
205     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
206     @post __post.privateIssuedSale == privateIssuedSale + _value * E18
207  */

```

Line 208-225 in File VALLIXToken.sol

```

208  function privateIssue(address _to, uint _value) onlyIssuer public
209  {
210      uint tokens = _value * E18;
211      require(maxSaleSupply >= tokenIssuedSale.add(tokens));
212
213      balances[_to] = balances[_to].add( tokens.mul(10)/100 );
214      privateFirstWallet[_to] = privateFirstWallet[_to].add( tokens.mul(10)/100
215      );
216      privateSecondWallet[_to] = privateSecondWallet[_to].add( tokens.mul(10)/100
217      );
218      privateThirdWallet[_to] = privateThirdWallet[_to].add( tokens.mul(20)/100
219      );
220      privateFourthWallet[_to] = privateFourthWallet[_to].add( tokens.mul(20)/100
221      );
222      privateFifthWallet[_to] = privateFifthWallet[_to].add( tokens.mul(30)/100
223      );
224
225      totalTokenSupply = totalTokenSupply.add(tokens);
226      tokenIssuedSale = tokenIssuedSale.add(tokens);
227      privateIssuedSale = privateIssuedSale.add(tokens);
228
229      emit SaleIssue(_to, tokens);
230  }

```

✓ The code meets the specification.

Formal Verification Request 8

publicIssue

📅 30, Aug 2019

🕒 1276.27 ms

Line 226-234 in File VALLIXToken.sol

```
226 /*@CTK publicIssue
227     @tag assume_completion
228     @post msg.sender == issuer
229     @post maxSaleSupply >= tokenIssuedSale + _value * E18
230     @post __post.balances[_to] == balances[_to] + _value * E18
231     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
232     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
233     @post __post.publicIssuedSale == publicIssuedSale + _value * E18
234 */
```

Line 235-247 in File VALLIXToken.sol

```
235 function publicIssue(address _to, uint _value) onlyIssuer public
236 {
237     uint tokens = _value * E18;
238     require(maxSaleSupply >= tokenIssuedSale.add(tokens));
239
240     balances[_to] = balances[_to].add(tokens);
241
242     totalTokenSupply = totalTokenSupply.add(tokens);
243     tokenIssuedSale = tokenIssuedSale.add(tokens);
244     publicIssuedSale = publicIssuedSale.add(tokens);
245
246     emit SaleIssue(_to, tokens);
247 }
```

✅ The code meets the specification.

Formal Verification Request 9

crowdIssue

📅 30, Aug 2019

🕒 672.15 ms

Line 248-255 in File VALLIXToken.sol

```
248 /*@CTK crowdIssue
249     @tag assume_completion
250     @post msg.sender == issuer
251     @post maxCrowdSupply >= tokenIssuedCrowd + _value * E18
252     @post __post.balances[_to] == balances[_to] + _value * E18
253     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
254     @post __post.tokenIssuedCrowd == tokenIssuedCrowd + _value * E18
255 */
```

Line 256-267 in File VALLIXToken.sol


```

256 function crowdIssue(address _to, uint _value) onlyIssuer public
257 {
258     uint tokens = _value * E18;
259     require(maxCrowdSupply >= tokenIssuedCrowd.add(tokens));
260
261     balances[_to] = balances[_to].add(tokens);
262
263     totalTokenSupply = totalTokenSupply.add(tokens);
264     tokenIssuedCrowd = tokenIssuedCrowd.add(tokens);
265
266     emit CrowdIssue(_to, tokens);
267 }

```

✓ The code meets the specification.

Formal Verification Request 10

MktIssue

📅 30, Aug 2019

🕒 561.65 ms

Line 269-276 in File VALLIXToken.sol

```

269 /*@CTK MktIssue
270     @tag assume_completion
271     @post msg.sender == issuer
272     @post maxMktSupply >= tokenIssuedMkt + _value * E18
273     @post __post.balances[_to] == balances[_to] + _value * E18
274     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
275     @post __post.tokenIssuedMkt == tokenIssuedMkt + _value * E18
276 */

```

Line 277-288 in File VALLIXToken.sol

```

277 function mktIssue(address _to, uint _value) onlyIssuer public
278 {
279     uint tokens = _value * E18;
280     require(maxMktSupply >= tokenIssuedMkt.add(tokens));
281
282     balances[_to] = balances[_to].add(tokens);
283
284     totalTokenSupply = totalTokenSupply.add(tokens);
285     tokenIssuedMkt = tokenIssuedMkt.add(tokens);
286
287     emit MktIssue(_to, tokens);
288 }

```

✓ The code meets the specification.

Formal Verification Request 11

reserveIssue

📅 30, Aug 2019

🕒 688.47 ms

Line 289-296 in File VALLIXToken.sol

```

289  /*@CTK reserveIssue
290     @tag assume_completion
291     @post msg.sender == issuer
292     @post maxReserveSupply >= tokenIssuedReserve + _value * E18
293     @post __post.balances[_to] == balances[_to] + _value * E18
294     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
295     @post __post.tokenIssuedReserve == tokenIssuedReserve + _value * E18
296  */

```

Line 297-308 in File VALLIXToken.sol

```

297  function reserveIssue(address _to, uint _value) onlyIssuer public
298  {
299      uint tokens = _value * E18;
300      require(maxReserveSupply >= tokenIssuedReserve.add(tokens));
301
302      balances[_to] = balances[_to].add(tokens);
303
304      totalTokenSupply = totalTokenSupply.add(tokens);
305      tokenIssuedReserve = tokenIssuedReserve.add(tokens);
306
307      emit ReserveIssue(_to, tokens);
308  }


```

✓ The code meets the specification.

Formal Verification Request 12

teamIssueVesting

 30, Aug 2019

 5064.28 ms

Line 309-321 in File VALLIXToken.sol

```

309  /*@CTK teamIssueVesting
310     @tag assume_completion
311     @post msg.sender == issuer
312     @post !saleTime
313     @post teamVestingTime >= _time
314     @post (endSaleTime + _time * teamVestingDate < now) &&
315           (teamVestingTimeAtSupply[_time] > 0)
316     @post maxTeamSupply >= tokenIssuedTeam + teamVestingTimeAtSupply[_time]
317     @post __post.balances[_to] == balances[_to] + teamVestingTimeAtSupply[_time]
318     @post __post.teamVestingTimeAtSupply[_time] == 0
319     @post __post.totalTokenSupply == totalTokenSupply + teamVestingTimeAtSupply[
320           _time]
321     @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingTimeAtSupply[_time]
322  */

```

Line 322-341 in File VALLIXToken.sol

```

322  function teamIssueVesting(address _to, uint _time) onlyIssuer public
323  {
324      require(saleTime == false);
325      require(teamVestingTime >= _time);

```

```

326
327     uint time = now;
328     require( ( ( endSaleTime + (_time * teamVestingDate) ) < time ) && (
329         teamVestingTimeAtSupply[_time] > 0 ));
330
331     uint tokens = teamVestingTimeAtSupply[_time];
332
333     require(maxTeamSupply >= tokenIssuedTeam.add(tokens));
334
335     balances[_to] = balances[_to].add(tokens);
336     teamVestingTimeAtSupply[_time] = 0;
337
338     totalTokenSupply = totalTokenSupply.add(tokens);
339     tokenIssuedTeam = tokenIssuedTeam.add(tokens);
340
341     emit TeamIssue(_to, tokens);
342 }

```

✓ The code meets the specification.

Formal Verification Request 13

advisorIssue

📅 30, Aug 2019

🕒 683.77 ms

Line 343-350 in File VALLIXToken.sol

```

343     /*CTK advisorIssue
344         @tag assume_completion
345         @post msg.sender == issuer
346         @post maxAdvisorSupply >= tokenIssuedAdvisor + _value * E18
347         @post __post.balances[_to] == balances[_to] + _value * E18
348         @post __post.totalTokenSupply == totalTokenSupply + _value * E18
349         @post __post.tokenIssuedAdvisor == tokenIssuedAdvisor + _value * E18
350     */

```

Line 351-363 in File VALLIXToken.sol

```

351     function advisorIssue(address _to, uint _value) onlyIssuer public
352     {
353         uint tokens = _value * E18;
354
355         require(maxAdvisorSupply >= tokenIssuedAdvisor.add(tokens));
356
357         balances[_to] = balances[_to].add(tokens);
358
359         totalTokenSupply = totalTokenSupply.add(tokens);
360         tokenIssuedAdvisor = tokenIssuedAdvisor.add(tokens);
361
362         emit AdvisorIssue(_to, tokens);
363     }


```

✓ The code meets the specification.

Formal Verification Request 14

isTransferable

 30, Aug 2019

 3.64 ms

Line 364-367 in File VALLIXToken.sol

```
364  /*@CTK isTransferable
365      @post !tokenLock || msg.sender == manager -> __return
366      @post !__return -> tokenLock && msg.sender != manager
367  */
```

Line 368-380 in File VALLIXToken.sol


```
368  function isTransferable() private view returns (bool)
369  {
370      if(tokenLock == false)
371      {
372          return true;
373      }
374      else if(msg.sender == manager)
375      {
376          return true;
377      }
378
379      return false;
380  }
```

 The code meets the specification.

Formal Verification Request 15

setTokenUnlock

 30, Aug 2019

 51.63 ms

Line 382-388 in File VALLIXToken.sol

```
382  /*@CTK setTokenUnlock
383      @tag assume_completion
384      @post msg.sender == manager
385      @post tokenLock
386      @post !saleTime
387      @post !__post.tokenLock
388  */
```

Line 389-395 in File VALLIXToken.sol

```
389  function setTokenUnlock() onlyManager public
390  {
391      require(tokenLock == true);
392      require(saleTime == false);
393
394      tokenLock = false;
395  }
```

✓ The code meets the specification.

Formal Verification Request 16

setTokenLock

📅 30, Aug 2019

🕒 41.86 ms

Line 396-401 in File VALLIXToken.sol

```
396  /*@CTK setTokenLock
397      @tag assume_completion
398      @post msg.sender == manager
399      @post !tokenLock
400      @post __post.tokenLock
401  */
```

Line 402-407 in File VALLIXToken.sol

```
402  function setTokenLock() onlyManager public
403  {
404      require(tokenLock == false);
405
406      tokenLock = true;
407  }
```

✓ The code meets the specification.

Formal Verification Request 17

privateUnlock

📅 30, Aug 2019

🕒 765.62 ms

Line 409-414 in File VALLIXToken.sol

```
409  /*@CTK privateUnlock
410      @tag assume_completion
411      @post msg.sender == manager
412      @post !tokenLock
413      @post !saleTime
414  */
```

Line 415-459 in File VALLIXToken.sol

```
415  function privateUnlock(address _to) onlyManager public
416  {
417      require(tokenLock == false);
418      require(saleTime == false);
419
420      uint time = now;
421      uint unlockTokens = 0;
422
423      if( (time >= endSaleTime.add(month)) && (privateFirstWallet[_to] > 0) )
424      {
```

```

425         balances[_to] = balances[_to].add(privateFirstWallet[_to]);
426         unlockTokens = unlockTokens.add(privateFirstWallet[_to]);
427         privateFirstWallet[_to] = 0;
428     }
429
430     if( (time >= endSaleTime.add(month * 2)) && (privateSecondWallet[_to] > 0) )
431     {
432         balances[_to] = balances[_to].add(privateSecondWallet[_to]);
433         unlockTokens = unlockTokens.add(privateSecondWallet[_to]);
434         privateSecondWallet[_to] = 0;
435     }
436
437     if( (time >= endSaleTime.add(month * 3)) && (privateThirdWallet[_to] > 0) )
438     {
439         balances[_to] = balances[_to].add(privateThirdWallet[_to]);
440         unlockTokens = unlockTokens.add(privateThirdWallet[_to]);
441         privateThirdWallet[_to] = 0;
442     }
443
444     if( (time >= endSaleTime.add(month * 4)) && (privateFourthWallet[_to] > 0) )
445     {
446         balances[_to] = balances[_to].add(privateFourthWallet[_to]);
447         unlockTokens = unlockTokens.add(privateFourthWallet[_to]);
448         privateFourthWallet[_to] = 0;
449     }
450
451     if( (time >= endSaleTime.add(month * 5)) && (privateFifthWallet[_to] > 0) )
452     {
453         balances[_to] = balances[_to].add(privateFifthWallet[_to]);
454         unlockTokens = unlockTokens.add(privateFifthWallet[_to]);
455         privateFifthWallet[_to] = 0;
456     }
457
458     emit TokenUnLock(_to, unlockTokens);
459 }


```

✓ The code meets the specification.

Formal Verification Request 18

endSale

 30, Aug 2019

 56.54 ms

Line 466-471 in File VALLIXToken.sol

```

466     /*@CTK endSale
467         @tag assume_completion
468         @post saleTime == true
469         @post __post.saleTime == false
470         @post __post.endSaleTime == now
471     */

```

Line 472-489 in File VALLIXToken.sol

```

472     function endSale() onlyManager public
473     {

```

```

474     require(saleTime == true);
475
476     saleTime = false;
477
478     uint time = now;
479
480     endSaleTime = time;
481
482     /*@IGNORE
483     for(uint i = 1; i <= teamVestingTime; i++)
484     {
485         teamVestingTimeAtSupply[i] = teamVestingTimeAtSupply[i].add(
486             teamVestingSupplyPerTime);
487     }
488     @IGNORE*/
489 }

```

✓ The code meets the specification.

Formal Verification Request 19

burnToken

📅 30, Aug 2019

🕒 357.32 ms

Line 495-502 in File VALLIXToken.sol

```

495     /*@CTK burnToken
496     @tag assume_completion
497     @post msg.sender == manager
498     @post balances[msg.sender] >= _value * E18
499     @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
500     @post __post.burnTokenSupply == burnTokenSupply + _value * E18
501     @post __post.totalTokenSupply == totalTokenSupply - _value * E18
502     */

```

Line 503-515 in File VALLIXToken.sol

```

503     function burnToken(uint _value) onlyManager public
504     {
505         uint tokens = _value * E18;
506
507         require(balances[msg.sender] >= tokens);
508
509         balances[msg.sender] = balances[msg.sender].sub(tokens);
510
511         burnTokenSupply = burnTokenSupply.add(tokens);
512         totalTokenSupply = totalTokenSupply.sub(tokens);
513
514         emit Burn(msg.sender, tokens);
515     }

```

✓ The code meets the specification.

Formal Verification Request 20

SafeMath add

📅 30, Aug 2019

🕒 24.02 ms

Line 5-11 in File SafeMath.sol

```
5  /*@CTK "SafeMath add"
6    @post (a + b < a || a + b < b) == __reverted
7    @post !__reverted -> __return == a + b
8    @post !__reverted -> !__has_overflow
9    @post !__reverted -> !__has_assertion_failure
10   @post !(__has_buf_overflow)
11   */
```

Line 12-18 in File SafeMath.sol

```
12  function add(uint256 a, uint256 b) internal pure returns (uint256)
13  {
14      uint256 c = a + b;
15      require(c >= a, "SafeMath: addition overflow");
16
17      return c;
18  }
```

✅ The code meets the specification.

Formal Verification Request 21

SafeMath sub

📅 30, Aug 2019

🕒 19.9 ms

Line 19-25 in File SafeMath.sol

```
19  /*@CTK "SafeMath sub"
20    @post (a < b) == __reverted
21    @post !__reverted -> __return == a - b
22    @post !__reverted -> !__has_overflow
23    @post !__reverted -> !__has_assertion_failure
24    @post !(__has_buf_overflow)
25   */
```

Line 26-32 in File SafeMath.sol

```
26  function sub(uint256 a, uint256 b) internal pure returns (uint256)
27  {
28      require(b <= a, "SafeMath: subtraction overflow");
29      uint256 c = a - b;
30
31      return c;
32  }
```

✅ The code meets the specification.

Formal Verification Request 22

SafeMath mul

📅 30, Aug 2019

🕒 278.29 ms

Line 33-39 in File SafeMath.sol

```
33  /*@CTK "SafeMath mul"
34     @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
35     @post !__reverted -> __return == a * b
36     @post !__reverted == !__has_overflow
37     @post !__reverted -> !__has_assertion_failure
38     @post !(__has_buf_overflow)
39  */
```

Line 40-50 in File SafeMath.sol

```
40  function mul(uint256 a, uint256 b) internal pure returns (uint256)
41  {
42      if (a == 0) {
43          return 0;
44      }
45
46      uint256 c = a * b;
47      require(c / a == b, "SafeMath: multiplication overflow");
48
49      return c;
50  }
```

✅ The code meets the specification.

Formal Verification Request 23

SafeMath div

📅 30, Aug 2019

🕒 20.36 ms

Line 51-57 in File SafeMath.sol

```
51  /*@CTK "SafeMath div"
52     @post (b <= 0) == __reverted
53     @post !__reverted -> __return == a / b
54     @post !__reverted -> !__has_overflow
55     @post !__reverted -> !__has_assertion_failure
56     @post !(__has_buf_overflow)
57  */
```

Line 58-65 in File SafeMath.sol

```
58  function div(uint256 a, uint256 b) internal pure returns (uint256)
59  {
60      require(b > 0, "SafeMath: division by zero");
61      uint256 c = a / b;
62      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
63
64      return c;
```

65 }

✓ The code meets the specification.

Formal Verification Request 24

SafeMath mod

📅 30, Aug 2019

🕒 18.34 ms

Line 66-72 in File SafeMath.sol

```
66     /*@CTK "SafeMath mod"
67         @post (b == 0) == __reverted
68         @post !__reverted -> __return == a % b
69         @post !__reverted -> !__has_overflow
70         @post !__reverted -> !__has_assertion_failure
71         @post !(__has_buf_overflow)
72     */
```

Line 73-77 in File SafeMath.sol

```
73     function mod(uint256 a, uint256 b) internal pure returns (uint256)
74     {
75         require(b != 0, "SafeMath: modulo by zero");
76         return a % b;
77     }
```

✓ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

📅 30, Aug 2019

🕒 12.49 ms

Line 31 in File OwnerHelper.sol

```
31     /*@CTK NO_OVERFLOW
```

Line 37-40 in File OwnerHelper.sol


```
37     constructor() public
38     {
39         master = msg.sender;
40     }
```

✓ The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

 30, Aug 2019

 1.32 ms

Line 32 in File OwnerHelper.sol

32 `//@CTK NO_ASF`

Line 37-40 in File OwnerHelper.sol


```
37     constructor() public
38     {
39         master = msg.sender;
40     }
```

 The code meets the specification.

Formal Verification Request 27

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

 30, Aug 2019

 1.02 ms

Line 33 in File OwnerHelper.sol

33 `//@CTK NO_BUF_OVERFLOW`

Line 37-40 in File OwnerHelper.sol


```
37     constructor() public
38     {
39         master = msg.sender;
40     }
```

 The code meets the specification.

Formal Verification Request 28

OwnerHelper

 30, Aug 2019

 1.52 ms

Line 34-36 in File OwnerHelper.sol

```
34     /*@CTK "OwnerHelper"
35         @post __post.master == msg.sender
36     */
```

Line 37-40 in File OwnerHelper.sol

```
37     constructor() public
38     {
39         master = msg.sender;
40     }
```

✓ The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

📅 30, Aug 2019

🕒 58.73 ms

Line 42 in File OwnerHelper.sol

42 `//@CTK NO_OVERFLOW`

Line 54-65 in File OwnerHelper.sol

```
54 function transferMastership(address _to) onlyMaster public
55 {
56     require(_to != master);
57     require(_to != issuer);
58     require(_to != manager);
59     require(_to != address(0x0));
60
61     address from = master;
62     master = _to;
63
64     emit ChangeMaster(from, _to);
65 }
```

✓ The code meets the specification.

Formal Verification Request 30

Method will not encounter an assertion failure.

📅 30, Aug 2019

🕒 3.85 ms

Line 43 in File OwnerHelper.sol

43 `//@CTK NO_ASF`

Line 54-65 in File OwnerHelper.sol

```
54 function transferMastership(address _to) onlyMaster public
55 {
56     require(_to != master);
57     require(_to != issuer);
58     require(_to != manager);
59     require(_to != address(0x0));
60
61     address from = master;
62     master = _to;
63
64     emit ChangeMaster(from, _to);
65 }
```

✓ The code meets the specification.

Formal Verification Request 31

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

📅 30, Aug 2019

🕒 13.91 ms

Line 44 in File OwnerHelper.sol

44 `//@CTK NO_BUF_OVERFLOW`

Line 54-65 in File OwnerHelper.sol

```

54  function transferMastership(address _to) onlyMaster public
55  {
56      require(_to != master);
57      require(_to != issuer);
58      require(_to != manager);
59      require(_to != address(0x0));
60
61      address from = master;
62      master = _to;
63
64      emit ChangeMaster(from, _to);
65  }

```

✅ The code meets the specification.

Formal Verification Request 32

transferMastership in OwnerHelper

📅 30, Aug 2019

🕒 5.35 ms

Line 45-53 in File OwnerHelper.sol

```

45  /*@CTK "transferMastership in OwnerHelper"
46      @tag assume_completion
47      @post msg.sender == master
48      @post _to != master
49      @post _to != issuer
50      @post _to != manager
51      @post _to != address(0)
52      @post __post.master == _to
53  */

```

Line 54-65 in File OwnerHelper.sol

```

54  function transferMastership(address _to) onlyMaster public
55  {
56      require(_to != master);
57      require(_to != issuer);
58      require(_to != manager);
59      require(_to != address(0x0));
60
61      address from = master;
62      master = _to;
63

```

```
64     emit ChangeMaster(from, _to);
65 }
```

✓ The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

📅 30, Aug 2019

🕒 52.62 ms

Line 66 in File OwnerHelper.sol

```
66 // @CTK NO_OVERFLOW
```

Line 78-89 in File OwnerHelper.sol

```
78 function transferIssuer(address _to) onlyMaster public
79 {
80     require(_to != master);
81     require(_to != issuer);
82     require(_to != manager);
83     require(_to != address(0x0));
84
85     address from = issuer;
86     issuer = _to;
87
88     emit ChangeIssuer(from, _to);
89 }
```

✓ The code meets the specification.

Formal Verification Request 34

Method will not encounter an assertion failure.

📅 30, Aug 2019

🕒 4.05 ms

Line 67 in File OwnerHelper.sol

```
67 // @CTK NO_ASF
```

Line 78-89 in File OwnerHelper.sol

```
78 function transferIssuer(address _to) onlyMaster public
79 {
80     require(_to != master);
81     require(_to != issuer);
82     require(_to != manager);
83     require(_to != address(0x0));
84
85     address from = issuer;
86     issuer = _to;
87
88     emit ChangeIssuer(from, _to);
89 }
```

✓ The code meets the specification.

Formal Verification Request 35

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

📅 30, Aug 2019

🕒 3.85 ms

Line 68 in File OwnerHelper.sol

```
68 // @CTK NO_BUF_OVERFLOW
```

Line 78-89 in File OwnerHelper.sol

```
78 function transferIssuer(address _to) onlyMaster public
79 {
80     require(_to != master);
81     require(_to != issuer);
82     require(_to != manager);
83     require(_to != address(0x0));
84
85     address from = issuer;
86     issuer = _to;
87
88     emit ChangeIssuer(from, _to);
89 }
```

✓ The code meets the specification.

Formal Verification Request 36

transferIssuer in OwnerHelper

📅 30, Aug 2019

🕒 4.94 ms

Line 69-77 in File OwnerHelper.sol

```
69 /* @CTK "transferIssuer in OwnerHelper"
70     @tag assume_completion
71     @post msg.sender == master
72     @post _to != master
73     @post _to != issuer
74     @post _to != manager
75     @post _to != address(0)
76     @post __post.issuer == _to
77 */
```

Line 78-89 in File OwnerHelper.sol

```
78 function transferIssuer(address _to) onlyMaster public
79 {
80     require(_to != master);
81     require(_to != issuer);
82     require(_to != manager);
83     require(_to != address(0x0));
```

```
84
85     address from = issuer;
86     issuer = _to;
87
88     emit ChangeIssuer(from, _to);
89 }
```

✓ The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

📅 30, Aug 2019

🕒 52.41 ms

Line 90 in File OwnerHelper.sol

```
90     //@CTK NO_OVERFLOW
```

Line 102-113 in File OwnerHelper.sol

```
102     function transferManager(address _to) onlyMaster public
103     {
104         require(_to != master);
105         require(_to != issuer);
106         require(_to != manager);
107         require(_to != address(0x0));
108
109         address from = manager;
110         manager = _to;
111
112         emit ChangeManager(from, _to);
113     }
```

✓ The code meets the specification.

Formal Verification Request 38

Method will not encounter an assertion failure.

📅 30, Aug 2019

🕒 3.95 ms

Line 91 in File OwnerHelper.sol

```
91     //@CTK NO_ASF
```

Line 102-113 in File OwnerHelper.sol

```
102     function transferManager(address _to) onlyMaster public
103     {
104         require(_to != master);
105         require(_to != issuer);
106         require(_to != manager);
107         require(_to != address(0x0));
108     }
```



```

109     address from = manager;
110     manager = _to;
111
112     emit ChangeManager(from, _to);
113 }

```

✓ The code meets the specification.

Formal Verification Request 39

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

📅 30, Aug 2019

🕒 3.98 ms

Line 92 in File OwnerHelper.sol

```
92  //@CTK NO_BUF_OVERFLOW
```

Line 102-113 in File OwnerHelper.sol

```

102  function transferManager(address _to) onlyMaster public
103  {
104      require(_to != master);
105      require(_to != issuer);
106      require(_to != manager);
107      require(_to != address(0x0));
108
109      address from = manager;
110      manager = _to;
111
112      emit ChangeManager(from, _to);
113  }

```

✓ The code meets the specification.

Formal Verification Request 40

transferManager in OwnerHelper

📅 30, Aug 2019

🕒 4.97 ms

Line 93-101 in File OwnerHelper.sol

```

93  /*@CTK "transferManager in OwnerHelper"
94      @tag assume_completion
95      @post msg.sender == master
96      @post _to != master
97      @post _to != issuer
98      @post _to != manager
99      @post _to != address(0)
100      @post __post.manager == _to
101  */

```

Line 102-113 in File OwnerHelper.sol

```
102 function transferManager(address _to) onlyMaster public
103 {
104     require(_to != master);
105     require(_to != issuer);
106     require(_to != manager);
107     require(_to != address(0x0));
108
109     address from = manager;
110     manager = _to;
111
112     emit ChangeManager(from, _to);
113 }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File VALLIXToken.sol

```

1  pragma solidity ^0.5.0;
2
3  import "./ERC20Interface.sol";
4  import "./OwnerHelper.sol";
5  import "./SafeMath.sol";
6
7  contract VALLIXToken is ERC20Interface, OwnerHelper
8  {
9      using SafeMath for uint;
10
11     string public name;
12     uint public decimals;
13     string public symbol;
14
15     uint constant private E18 = 10000000000000000000;
16     uint constant private month = 2592000;
17
18     uint constant public maxTotalSupply = 10000000000 * E18;
19
20     uint constant public maxSaleSupply = 2000000000 * E18;
21     uint constant public maxCrowdSupply = 1600000000 * E18;
22     uint constant public maxMktSupply = 2800000000 * E18;
23     uint constant public maxTeamSupply = 1600000000 * E18;
24     uint constant public maxReserveSupply = 1600000000 * E18;
25     uint constant public maxAdvisorSupply = 400000000 * E18;
26
27     uint constant public teamVestingSupplyPerTime = 100000000 * E18;
28     uint constant public teamVestingDate = 2 * month;
29     uint constant public teamVestingTime = 16;
30
31     uint public totalTokenSupply;
32
33     uint public tokenIssuedSale;
34     uint public privateIssuedSale;
35     uint public publicIssuedSale;
36     uint public tokenIssuedCrowd;
37     uint public tokenIssuedMkt;
38     uint public tokenIssuedTeam;
39     uint public tokenIssuedReserve;
40     uint public tokenIssuedAdvisor;
41
42     uint public burnTokenSupply;
43
44     mapping (address => uint) public balances;
45     mapping (address => mapping ( address => uint )) public approvals;
46
47     mapping (address => uint) public privateFirstWallet;
48     mapping (address => uint) public privateSecondWallet;
49     mapping (address => uint) public privateThirdWallet;
50     mapping (address => uint) public privateFourthWallet;
51     mapping (address => uint) public privateFifthWallet;
52
53     mapping (uint => uint) public teamVestingTimeAtSupply;
54

```

```

55  bool public tokenLock = true;
56  bool public saleTime = true;
57  uint public endSaleTime = 0;
58
59  event Burn(address indexed _from, uint _value);
60
61  event SaleIssue(address indexed _to, uint _tokens);
62  event CrowdIssue(address indexed _to, uint _tokens);
63  event MktIssue(address indexed _to, uint _tokens);
64  event TeamIssue(address indexed _to, uint _tokens);
65  event ReserveIssue(address indexed _to, uint _tokens);
66  event AdvisorIssue(address indexed _to, uint _tokens);
67
68  event TokenUnLock(address indexed _to, uint _tokens);
69
70  /*CTK constructor
71   @tag assume_completion
72   @post __post.name == "VALLIX Token"
73   @post __post.decimals == 18
74   @post __post.symbol == "VLX"
75   @post __post.totalTokenSupply == 0
76   @post __post.tokenIssuedSale == 0
77   @post __post.tokenIssuedCrowd == 0
78   @post __post.tokenIssuedMkt == 0
79   @post __post.tokenIssuedTeam == 0
80   @post __post.tokenIssuedReserve == 0
81   @post __post.tokenIssuedAdvisor == 0
82
83   @post maxTotalSupply == maxSaleSupply + maxCrowdSupply + maxMktSupply +
      maxTeamSupply + maxReserveSupply + maxAdvisorSupply
84   @post maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime
85  */
86  constructor() public
87  {
88      name      = "VALLIX Token";
89      decimals  = 18;
90      symbol    = "VLX";
91
92      totalTokenSupply = 0;
93
94      tokenIssuedSale    = 0;
95      tokenIssuedCrowd   = 0;
96      tokenIssuedMkt     = 0;
97      tokenIssuedTeam    = 0;
98      tokenIssuedReserve = 0;
99      tokenIssuedAdvisor = 0;
100
101      require(maxTotalSupply == maxSaleSupply + maxCrowdSupply + maxMktSupply +
        maxTeamSupply + maxReserveSupply + maxAdvisorSupply);
102
103      require(maxTeamSupply == teamVestingSupplyPerTime * teamVestingTime);
104
105  }
106
107  /*@CTK totalSupply
108   @post __return == totalTokenSupply
109  */
110  function totalSupply() view public returns (uint)

```

```

111 {
112     return totalTokenSupply;
113 }
114 /*@CTK balanceOf
115    @tag assume_completion
116    @post __return == (balances[_who] + privateFirstWallet[_who] +
117        privateSecondWallet[_who] + privateThirdWallet[_who] + privateFourthWallet[
118            _who] + privateFifthWallet[_who])
119 */
120 function balanceOf(address _who) view public returns (uint)
121 {
122     uint balance = balances[_who];
123     balance = balance.add(privateFirstWallet[_who] + privateSecondWallet[_who] +
124         privateThirdWallet[_who] + privateFourthWallet[_who] + privateFifthWallet[
125             _who]);
126     return balance;
127 }
128
129 /*@CTK transfer
130    @tag assume_completion
131    @pre msg.sender != _to
132    @post balances[msg.sender] >= _value
133    @post __post.balances[msg.sender] == balances[msg.sender] - _value
134    @post __post.balances[_to] == balances[_to] + _value
135 */
136 function transfer(address _to, uint _value) public returns (bool)
137 {
138     require(isTransferable() == true);
139     require(balances[msg.sender] >= _value);
140
141     balances[msg.sender] = balances[msg.sender].sub(_value);
142     balances[_to] = balances[_to].add(_value);
143
144     emit Transfer(msg.sender, _to, _value);
145
146     return true;
147 }
148
149 /*@CTK approve
150    @tag assume_completion
151    @pre msg.sender != _spender
152    @post balances[msg.sender] >= _value
153    @post __post.approvals[msg.sender][_spender] == _value
154 */
155 function approve(address _spender, uint _value) public returns (bool)
156 {
157     require(isTransferable() == true);
158     require(balances[msg.sender] >= _value);
159
160     approvals[msg.sender][_spender] = _value;
161
162     emit Approval(msg.sender, _spender, _value);
163
164     return true;
165 }
166
167 /*@CTK allowance

```

```

165     @post __return == approvals[_owner][_spender]
166     */
167     function allowance(address _owner, address _spender) view public returns (uint)
168     {
169         return approvals[_owner][_spender];
170     }
171
172     /*@CTK transferFrom
173     @tag assume_completion
174     @pre _from != _to
175     @post balances[_from] >= _value
176     @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
177         _value
178     @post __post.balances[_from] == balances[_from] - _value
179     @post __post.balances[_to] == balances[_to] + _value
180     */
181     function transferFrom(address _from, address _to, uint _value) public returns (
182         bool)
183     {
184         require(isTransferable() == true);
185         require(balances[_from] >= _value);
186         require(approvals[_from][msg.sender] >= _value);
187
188         approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
189         balances[_from] = balances[_from].sub(_value);
190         balances[_to] = balances[_to].add(_value);
191
192         emit Transfer(_from, _to, _value);
193
194         return true;
195     }
196
197     /*@CTK privateIssue
198     @tag assume_completion
199     @post msg.sender == issuer
200     @post maxSaleSupply >= tokenIssuedSale + (_value * E18)
201     @post __post.balances[_to] == balances[_to] + _value * E18 * 10 / 100
202     @post __post.privateFirstWallet[_to] == privateFirstWallet[_to] + _value * E18 *
203         10 / 100
204     @post __post.privateSecondWallet[_to] == privateSecondWallet[_to] + _value * E18
205         * 10 / 100
206     @post __post.privateThirdWallet[_to] == privateThirdWallet[_to] + _value * E18 *
207         20 / 100
208     @post __post.privateFourthWallet[_to] == privateFourthWallet[_to] + _value * E18
209         * 20 / 100
210     @post __post.privateFifthWallet[_to] == privateFifthWallet[_to] + _value * E18 *
211         30 / 100
212     @post __post.totalTokenSupply == totalTokenSupply + _value * E18
213     @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
214     @post __post.privateIssuedSale == privateIssuedSale + _value * E18
215     */
216     function privateIssue(address _to, uint _value) onlyIssuer public
217     {
218         uint tokens = _value * E18;
219         require(maxSaleSupply >= tokenIssuedSale.add(tokens));
220
221         balances[_to] = balances[_to].add( tokens.mul(10)/100 );
222         privateFirstWallet[_to] = privateFirstWallet[_to].add( tokens.mul(10)/100
223             );

```

```

215     privateSecondWallet[_to]      = privateSecondWallet[_to].add( tokens.mul(10)/100
216         );
217     privateThirdWallet[_to]       = privateThirdWallet[_to].add( tokens.mul(20)/100
218         );
219     privateFourthWallet[_to]      = privateFourthWallet[_to].add( tokens.mul(20)/100
220         );
221     privateFifthWallet[_to]       = privateFifthWallet[_to].add( tokens.mul(30)/100
222         );
223
224     totalTokenSupply = totalTokenSupply.add(tokens);
225     tokenIssuedSale = tokenIssuedSale.add(tokens);
226     privateIssuedSale = privateIssuedSale.add(tokens);
227
228     emit SaleIssue(_to, tokens);
229 }
230
231 /*@CTK publicIssue
232   @tag assume_completion
233   @post msg.sender == issuer
234   @post maxSaleSupply >= tokenIssuedSale + _value * E18
235   @post __post.balances[_to] == balances[_to] + _value * E18
236   @post __post.totalTokenSupply == totalTokenSupply + _value * E18
237   @post __post.tokenIssuedSale == tokenIssuedSale + _value * E18
238   @post __post.publicIssuedSale == publicIssuedSale + _value * E18
239 */
240 function publicIssue(address _to, uint _value) onlyIssuer public
241 {
242     uint tokens = _value * E18;
243     require(maxSaleSupply >= tokenIssuedSale.add(tokens));
244
245     balances[_to] = balances[_to].add(tokens);
246
247     totalTokenSupply = totalTokenSupply.add(tokens);
248     tokenIssuedSale = tokenIssuedSale.add(tokens);
249     publicIssuedSale = publicIssuedSale.add(tokens);
250
251     emit SaleIssue(_to, tokens);
252 }
253
254 /*@CTK crowdIssue
255   @tag assume_completion
256   @post msg.sender == issuer
257   @post maxCrowdSupply >= tokenIssuedCrowd + _value * E18
258   @post __post.balances[_to] == balances[_to] + _value * E18
259   @post __post.totalTokenSupply == totalTokenSupply + _value * E18
260   @post __post.tokenIssuedCrowd == tokenIssuedCrowd + _value * E18
261 */
262 function crowdIssue(address _to, uint _value) onlyIssuer public
263 {
264     uint tokens = _value * E18;
265     require(maxCrowdSupply >= tokenIssuedCrowd.add(tokens));
266
267     balances[_to] = balances[_to].add(tokens);
268
269     totalTokenSupply = totalTokenSupply.add(tokens);
270     tokenIssuedCrowd = tokenIssuedCrowd.add(tokens);
271
272     emit CrowdIssue(_to, tokens);
273 }

```

```

269  /*@CTK MktIssue
270      @tag assume_completion
271      @post msg.sender == issuer
272      @post maxMktSupply >= tokenIssuedMkt + _value * E18
273      @post __post.balances[_to] == balances[_to] + _value * E18
274      @post __post.totalTokenSupply == totalTokenSupply + _value * E18
275      @post __post.tokenIssuedMkt == tokenIssuedMkt + _value * E18
276  */
277  function mktIssue(address _to, uint _value) onlyIssuer public
278  {
279      uint tokens = _value * E18;
280      require(maxMktSupply >= tokenIssuedMkt.add(tokens));
281
282      balances[_to] = balances[_to].add(tokens);
283
284      totalTokenSupply = totalTokenSupply.add(tokens);
285      tokenIssuedMkt = tokenIssuedMkt.add(tokens);
286
287      emit MktIssue(_to, tokens);
288  }
289  /*@CTK reserveIssue
290      @tag assume_completion
291      @post msg.sender == issuer
292      @post maxReserveSupply >= tokenIssuedReserve + _value * E18
293      @post __post.balances[_to] == balances[_to] + _value * E18
294      @post __post.totalTokenSupply == totalTokenSupply + _value * E18
295      @post __post.tokenIssuedReserve == tokenIssuedReserve + _value * E18
296  */
297  function reserveIssue(address _to, uint _value) onlyIssuer public
298  {
299      uint tokens = _value * E18;
300      require(maxReserveSupply >= tokenIssuedReserve.add(tokens));
301
302      balances[_to] = balances[_to].add(tokens);
303
304      totalTokenSupply = totalTokenSupply.add(tokens);
305      tokenIssuedReserve = tokenIssuedReserve.add(tokens);
306
307      emit ReserveIssue(_to, tokens);
308  }
309  /*@CTK teamIssueVesting
310      @tag assume_completion
311      @post msg.sender == issuer
312      @post !saleTime
313      @post teamVestingTime >= _time
314      @post (endSaleTime + _time * teamVestingDate < now) &&
315              (teamVestingTimeAtSupply[_time] > 0)
316      @post maxTeamSupply >= tokenIssuedTeam + teamVestingTimeAtSupply[_time]
317      @post __post.balances[_to] == balances[_to] + teamVestingTimeAtSupply[_time]
318      @post __post.teamVestingTimeAtSupply[_time] == 0
319      @post __post.totalTokenSupply == totalTokenSupply + teamVestingTimeAtSupply[
320          _time]
321      @post __post.tokenIssuedTeam == tokenIssuedTeam + teamVestingTimeAtSupply[_time]
322  */
323  function teamIssueVesting(address _to, uint _time) onlyIssuer public
324  {
325      require(saleTime == false);
326      require(teamVestingTime >= _time);

```



```

326
327     uint time = now;
328     require( ( ( endSaleTime + (_time * teamVestingDate) ) < time ) && (
329         teamVestingTimeAtSupply[_time] > 0 ));
330
331     uint tokens = teamVestingTimeAtSupply[_time];
332
333     require(maxTeamSupply >= tokenIssuedTeam.add(tokens));
334
335     balances[_to] = balances[_to].add(tokens);
336     teamVestingTimeAtSupply[_time] = 0;
337
338     totalTokenSupply = totalTokenSupply.add(tokens);
339     tokenIssuedTeam = tokenIssuedTeam.add(tokens);
340
341     emit TeamIssue(_to, tokens);
342 }
343
344 /*@CTK advisorIssue
345 @tag assume_completion
346 @post msg.sender == issuer
347 @post maxAdvisorSupply >= tokenIssuedAdvisor + _value * E18
348 @post __post.balances[_to] == balances[_to] + _value * E18
349 @post __post.totalTokenSupply == totalTokenSupply + _value * E18
350 @post __post.tokenIssuedAdvisor == tokenIssuedAdvisor + _value * E18
351 */
352 function advisorIssue(address _to, uint _value) onlyIssuer public
353 {
354     uint tokens = _value * E18;
355
356     require(maxAdvisorSupply >= tokenIssuedAdvisor.add(tokens));
357
358     balances[_to] = balances[_to].add(tokens);
359
360     totalTokenSupply = totalTokenSupply.add(tokens);
361     tokenIssuedAdvisor = tokenIssuedAdvisor.add(tokens);
362
363     emit AdvisorIssue(_to, tokens);
364 }
365
366 /*@CTK isTransferable
367 @post !tokenLock || msg.sender == manager -> __return
368 @post !__return -> tokenLock && msg.sender != manager
369 */
370 function isTransferable() private view returns (bool)
371 {
372     if(tokenLock == false)
373     {
374         return true;
375     }
376     else if(msg.sender == manager)
377     {
378         return true;
379     }
380
381     return false;
382 }
383
384 /*@CTK setTokenUnlock

```

```

383     @tag assume_completion
384     @post msg.sender == manager
385     @post tokenLock
386     @post !saleTime
387     @post !__post.tokenLock
388     */
389     function setTokenUnlock() onlyManager public
390     {
391         require(tokenLock == true);
392         require(saleTime == false);
393
394         tokenLock = false;
395     }
396     /*@CTK setTokenLock
397     @tag assume_completion
398     @post msg.sender == manager
399     @post !tokenLock
400     @post __post.tokenLock
401     */
402     function setTokenLock() onlyManager public
403     {
404         require(tokenLock == false);
405
406         tokenLock = true;
407     }
408
409     /*@CTK privateUnlock
410     @tag assume_completion
411     @post msg.sender == manager
412     @post !tokenLock
413     @post !saleTime
414     */
415     function privateUnlock(address _to) onlyManager public
416     {
417         require(tokenLock == false);
418         require(saleTime == false);
419
420         uint time = now;
421         uint unlockTokens = 0;
422
423         if( (time >= endSaleTime.add(month)) && (privateFirstWallet[_to] > 0) )
424         {
425             balances[_to] = balances[_to].add(privateFirstWallet[_to]);
426             unlockTokens = unlockTokens.add(privateFirstWallet[_to]);
427             privateFirstWallet[_to] = 0;
428         }
429
430         if( (time >= endSaleTime.add(month * 2)) && (privateSecondWallet[_to] > 0) )
431         {
432             balances[_to] = balances[_to].add(privateSecondWallet[_to]);
433             unlockTokens = unlockTokens.add(privateSecondWallet[_to]);
434             privateSecondWallet[_to] = 0;
435         }
436
437         if( (time >= endSaleTime.add(month * 3)) && (privateThirdWallet[_to] > 0) )
438         {
439             balances[_to] = balances[_to].add(privateThirdWallet[_to]);
440             unlockTokens = unlockTokens.add(privateThirdWallet[_to]);

```

```

441     privateThirdWallet[_to] = 0;
442 }
443
444 if( (time >= endSaleTime.add(month * 4)) && (privateFourthWallet[_to] > 0) )
445 {
446     balances[_to] = balances[_to].add(privateFourthWallet[_to]);
447     unlockTokens = unlockTokens.add(privateFourthWallet[_to]);
448     privateFourthWallet[_to] = 0;
449 }
450
451 if( (time >= endSaleTime.add(month * 5)) && (privateFifthWallet[_to] > 0) )
452 {
453     balances[_to] = balances[_to].add(privateFifthWallet[_to]);
454     unlockTokens = unlockTokens.add(privateFifthWallet[_to]);
455     privateFifthWallet[_to] = 0;
456 }
457
458 emit TokenUnLock(_to, unlockTokens);
459 }
460
461 function () payable external
462 {
463     revert();
464 }
465
466 /*@CTK endSale
467    @tag assume_completion
468    @post saleTime == true
469    @post __post.saleTime == false
470    @post __post.endSaleTime == now
471 */
472 function endSale() onlyManager public
473 {
474     require(saleTime == true);
475
476     saleTime = false;
477
478     uint time = now;
479
480     endSaleTime = time;
481
482     for(uint i = 1; i <= teamVestingTime; i++)
483     {
484         teamVestingTimeAtSupply[i] = teamVestingTimeAtSupply[i].add(
485             teamVestingSupplyPerTime);
486     }
487 }
488
489 function transferAnyERC20Token(address tokenAddress, uint tokens) onlyManager
490     public returns (bool success)
491 {
492     return ERC20Interface(tokenAddress).transfer(manager, tokens);
493 }
494
495 /*@CTK burnToken
496    @tag assume_completion
497    @post msg.sender == manager
498    @post balances[msg.sender] >= _value * E18

```

```

497     @post __post.balances[msg.sender] == balances[msg.sender] - _value * E18
498     @post __post.burnTokenSupply == burnTokenSupply + _value * E18
499     @post __post.totalTokenSupply == totalTokenSupply - _value * E18
500     */
501     function burnToken(uint _value) onlyManager public
502     {
503         uint tokens = _value * E18;
504
505         require(balances[msg.sender] >= tokens);
506
507         balances[msg.sender] = balances[msg.sender].sub(tokens);
508
509         burnTokenSupply = burnTokenSupply.add(tokens);
510         totalTokenSupply = totalTokenSupply.sub(tokens);
511
512         emit Burn(msg.sender, tokens);
513     }
514
515     function close() onlyMaster public
516     {
517         selfdestruct(msg.sender);
518     }
519 }

```

File SafeMath.sol

```

1  pragma solidity ^0.5.0;
2
3  library SafeMath
4  {
5      /*@CTK "SafeMath add"
6          @post (a + b < a || a + b < b) == __reverted
7          @post !__reverted -> __return == a + b
8          @post !__reverted -> !__has_overflow
9          @post !__reverted -> !__has_assertion_failure
10         @post !(__has_buf_overflow)
11     */
12     function add(uint256 a, uint256 b) internal pure returns (uint256)
13     {
14         uint256 c = a + b;
15         require(c >= a, "SafeMath: addition overflow");
16
17         return c;
18     }
19     /*@CTK "SafeMath sub"
20         @post (a < b) == __reverted
21         @post !__reverted -> __return == a - b
22         @post !__reverted -> !__has_overflow
23         @post !__reverted -> !__has_assertion_failure
24         @post !(__has_buf_overflow)
25     */
26     function sub(uint256 a, uint256 b) internal pure returns (uint256)
27     {
28         require(b <= a, "SafeMath: subtraction overflow");
29         uint256 c = a - b;
30
31         return c;
32     }
33     /*@CTK "SafeMath mul"

```

```

34     @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
35     @post !__reverted -> __return == a * b
36     @post !__reverted == !__has_overflow
37     @post !__reverted -> !__has_assertion_failure
38     @post !(__has_buf_overflow)
39     */
40     function mul(uint256 a, uint256 b) internal pure returns (uint256)
41     {
42         if (a == 0) {
43             return 0;
44         }
45
46         uint256 c = a * b;
47         require(c / a == b, "SafeMath: multiplication overflow");
48
49         return c;
50     }
51     /*@CTK "SafeMath div"
52     @post (b <= 0) == __reverted
53     @post !__reverted -> __return == a / b
54     @post !__reverted -> !__has_overflow
55     @post !__reverted -> !__has_assertion_failure
56     @post !(__has_buf_overflow)
57     */
58     function div(uint256 a, uint256 b) internal pure returns (uint256)
59     {
60         require(b > 0, "SafeMath: division by zero");
61         uint256 c = a / b;
62         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
63
64         return c;
65     }
66     /*@CTK "SafeMath mod"
67     @post (b == 0) == __reverted
68     @post !__reverted -> __return == a % b
69     @post !__reverted -> !__has_overflow
70     @post !__reverted -> !__has_assertion_failure
71     @post !(__has_buf_overflow)
72     */
73     function mod(uint256 a, uint256 b) internal pure returns (uint256)
74     {
75         require(b != 0, "SafeMath: modulo by zero");
76         return a % b;
77     }
78 }

```

File OwnerHelper.sol

```

1  pragma solidity ^0.5.0;
2
3  contract OwnerHelper
4  {
5
6      address public master;
7      address public issuer;
8      address public manager;
9
10     event ChangeMaster(address indexed _from, address indexed _to);
11     event ChangeIssuer(address indexed _from, address indexed _to);

```

```

12  event ChangeManager(address indexed _from, address indexed _to);
13
14  modifier onlyMaster
15  {
16      require(msg.sender == master);
17      _;
18  }
19
20  modifier onlyIssuer
21  {
22      require(msg.sender == issuer);
23      _;
24  }
25
26  modifier onlyManager
27  {
28      require(msg.sender == manager);
29      _;
30  }
31  //@CTK NO_OVERFLOW
32  //@CTK NO_ASF
33  //@CTK NO_BUF_OVERFLOW
34  /*@CTK "OwnerHelper"
35      @post __post.master == msg.sender
36  */
37  constructor() public
38  {
39      master = msg.sender;
40  }
41
42  //@CTK NO_OVERFLOW
43  //@CTK NO_ASF
44  //@CTK NO_BUF_OVERFLOW
45  /*@CTK "transferMastership in OwnerHelper"
46      @tag assume_completion
47      @post msg.sender == master
48      @post _to != master
49      @post _to != issuer
50      @post _to != manager
51      @post _to != address(0)
52      @post __post.master == _to
53  */
54  function transferMastership(address _to) onlyMaster public
55  {
56      require(_to != master);
57      require(_to != issuer);
58      require(_to != manager);
59      require(_to != address(0x0));
60
61      address from = master;
62      master = _to;
63
64      emit ChangeMaster(from, _to);
65  }
66  //@CTK NO_OVERFLOW
67  //@CTK NO_ASF
68  //@CTK NO_BUF_OVERFLOW
69  /*@CTK "transferIssuer in OwnerHelper"

```

```
70     @tag assume_completion
71     @post msg.sender == master
72     @post _to != master
73     @post _to != issuer
74     @post _to != manager
75     @post _to != address(0)
76     @post _post.issuer == _to
77     */
78     function transferIssuer(address _to) onlyMaster public
79     {
80         require(_to != master);
81         require(_to != issuer);
82         require(_to != manager);
83         require(_to != address(0x0));
84
85         address from = issuer;
86         issuer = _to;
87
88         emit ChangeIssuer(from, _to);
89     }
90     //@CTK NO_OVERFLOW
91     //@CTK NO_ASF
92     //@CTK NO_BUF_OVERFLOW
93     /*@CTK "transferManager in OwnerHelper"
94         @tag assume_completion
95         @post msg.sender == master
96         @post _to != master
97         @post _to != issuer
98         @post _to != manager
99         @post _to != address(0)
100         @post _post.manager == _to
101     */
102     function transferManager(address _to) onlyMaster public
103     {
104         require(_to != master);
105         require(_to != issuer);
106         require(_to != manager);
107         require(_to != address(0x0));
108
109         address from = manager;
110         manager = _to;
111
112         emit ChangeManager(from, _to);
113     }
114 }
```