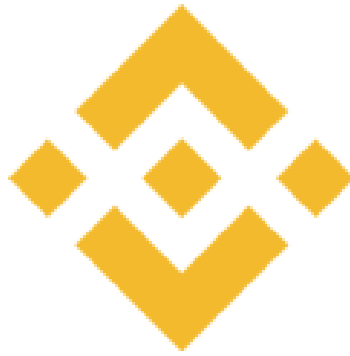


CERTIK AUDIT REPORT FOR BINANCE(BGBP)



Request Date: 2019-07-09
Revision Date: 2019-07-31
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	22

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 Binance(BGBP)(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 Binance(BGBP). This audit was conducted to discover issues and vulnerabilities in the source code of Binance(BGBP)'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.

Jul 31, 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.	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

- **BGBP.sol**
bd55853b8f4a53f2abe695695906f49d058fb4d92cd835465f7f3c8b8ad636c4
- **Migrations.sol**
ff98ae1d2490675979a38b1e5fe5bcf6b9b7f9f2b36dc3a3f8e1f2531bd9eb41
- **TestUpgradeBGBP.sol**
7fba4ba3205ee08273e42ef675b9e68f87c68bedaed04ac39174cfedc3aab40a

Summary

CertiK was chosen by Binance to audit the design and implementation of its soon to be released BGBP 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. The entries are labeled **CRITICAL** **IMPORTANT** **INFO** **DISCUSSION** (in decreasing significance level).

BGBP.sol previous version

- **IMPORTANT** **SafeMath, Ownable, BasicToken, StandardToken**: Usage of locally-included outdated contracts/libraries.
 - (Binance - Resolved) Use latest `openzeppelin-solidity` library.
- **IMPORTANT** **destroyBlackFunds()**: Recommend checking if `_blackListedUser` is a zero address. According to [etherscan](#), `address(0)` is currently holding ~ 7778 ether at the time of this report.
 - (Binance - Resolved) Check added in latest version.

- **INFO** `constructor()`: The `decimals` is defined as `uint`, which is the alias of `uint256`. According to the EIP20 specification `decimals` is using `uint8`. Consider changing it to comply with EIP20.
 - (Binance - Resolved) Changed data type of `decimals` from `uint` to `uint8`.
- **INFO** `transfer()`, `transferFrom()`: Only `msg.sender` is checked against the Black-Listed. Is Binance allowing to sender to transfer value to a blockListed address? If not, recommend adding check `require(!isBlackListed[_to], ...)`.
 - (Binance - Resolved) Check added in latest version.
- **INFO** `approve()`: When the token is under pause status, is it perform to `approve`? The current `approve()` allows any user to change the allowance while the token is paused. Shall the paused token halt all operations, except for those functions use in an emergency? If not, recommend to add the modifier `whenNotPaused`.
 - (Binance - Resolved) `whenNotPaused` added in latest version.
- **INFO** Consider to add `indexed` in the following events, which allows the client to track the status of blacklist.


```
event DestroyedBlackFunds(address indexed _blackListedUser, uint _balance);
event AddedBlackList(address indexed _user);
event RemovedBlackList(address indexed _user);
```

 - (Binance - Resolved) `indexed` added in latest version.
- **INFO** `deprecate()`: Highly recommend to prevent `_upgradedAddress` from being a zero address: `require(_upgradedAddress != address(0), ...)`
 - (Binance - Resolved) Check added in latest version.
- **DISCUSSION** `deprecate()`, `issue()`, `redeem()`: Recommend adding error messages to `require()` checks.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File BGBP.sol

```
1 pragma solidity 0.5.8;
```



⚠ Version to compile has the following bug: 0.5.8: DynamicConstructorArgumentsClipped-ABIV2

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

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

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

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


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

Counterexample	 This code violates the specification	
Initial environment	1	Counter Example:
	2	Before Execution:
	3	Input = {
	4	from = 0x0
	5	to = 0x0
	6	tokens = 0x6c
	7	}
	8	This = 0
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Formal Verification Request 1

getBlackListStatus

 31, Jul 2019

 8.99 ms

Line 10-13 in File BGBP.sol

```
10  /*@CTK getBlackListStatus
11     @tag assume_completion
12     @post __return == isBlackListed[_maker]
13  */
```

Line 14-16 in File BGBP.sol


```
14  function getBlackListStatus(address _maker) external view returns (bool) {
15      return isBlackListed[_maker];
16  }
```

 The code meets the specification.

Formal Verification Request 2

addBlackList

 31, Jul 2019

 63.15 ms

Line 20-25 in File BGBP.sol

```
20  /*@CTK addBlackList
21     @tag assume_completion
22     @post msg.sender == _owner
23     @post isBlackListed[_evilUser] == false
24     @post __post.isBlackListed[_evilUser] == true
25  */
```

Line 26-31 in File BGBP.sol


```
26  function addBlackList(address _evilUser) public onlyOwner {
27      require(!isBlackListed[_evilUser], "_evilUser is already in black list");
28
29      isBlackListed[_evilUser] = true;
30      emit AddedBlackList(_evilUser);
31  }
```

 The code meets the specification.

Formal Verification Request 3

removeBlackList

 31, Jul 2019

 43.47 ms

Line 33-38 in File BGBP.sol

```

33  /*@CTK removeBlackList
34      @tag assume_completion
35      @post msg.sender == _owner
36      @post isBlackListed[_clearedUser] == true
37      @post __post.isBlackListed[_clearedUser] == false
38  */

```

Line 39-43 in File BGBP.sol

```

39  function removeBlackList(address _clearedUser) public onlyOwner {
40      require(isBlackListed[_clearedUser], "_clearedUser isn't in black list");
41      isBlackListed[_clearedUser] = false;
42      emit RemovedBlackList(_clearedUser);
43  }

```

✓ The code meets the specification.

Formal Verification Request 4

destroyBlackFunds



31, Jul 2019



289.95 ms

Line 45-52 in File BGBP.sol

```

45  /*@CTK destroyBlackFunds
46      @tag assume_completion
47      @post msg.sender == _owner
48      @post _blackListedUser != address(0)
49      @post isBlackListed[_blackListedUser] == true
50      @post __post._balances[_blackListedUser] == 0
51      @post __post._totalSupply == _totalSupply - _balances[_blackListedUser]
52  */

```

Line 53-60 in File BGBP.sol

```

53  function destroyBlackFunds(address _blackListedUser) public onlyOwner {
54      require(_blackListedUser != address(0x0), "_blackListedUser is the zero address");
55      require(isBlackListed[_blackListedUser], "_blackListedUser isn't in black list");
56
57      uint256 dirtyFunds = balanceOf(_blackListedUser);
58      super._burn(_blackListedUser, dirtyFunds);
59      emit DestroyedBlackFunds(_blackListedUser, dirtyFunds);
60  }

```

✓ The code meets the specification.

Formal Verification Request 5

BGBPToken



31, Jul 2019



199.54 ms

Line 95-103 in File BGBP.sol

```

95  /*@CTK BGBPToken
96      @tag assume_completion
97      @post __post.name == _name
98      @post __post.symbol == _symbol
99      @post __post.decimals == _decimals
100     @post __post.deprecated == false
101     @post __post._balances[msg.sender] == _balances[msg.sender] + _initialSupply
102     @post __post._totalSupply == _totalSupply + _initialSupply
103  */

```

Line 104-110 in File BGBP.sol

```

104     constructor(uint256 _initialSupply, string memory _name, string memory _symbol,
105                uint8 _decimals) public {
106         name = _name;
107         symbol = _symbol;
108         decimals = _decimals;
109         deprecated = false;
110         super._mint(msg.sender, _initialSupply);
111     }

```

✓ The code meets the specification.

Formal Verification Request 6

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 376.71 ms

Line 113 in File BGBP.sol

```

113  //@CTK NO_OVERFLOW

```

Line 130-142 in File BGBP.sol

```

130     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool
131         success) {
132         require(!isBlackListed[msg.sender], "can't transfer token from address in black
133             list");
134         require(!isBlackListed[_to], "can't transfer token to address in black list");
135         if (deprecated) {
136             success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.
137                 sender, _to, _value);
138             require(success, "failed to call upgraded contract");
139             return true;
140         } else {
141             return super.transfer(_to, _value);
142         }
143     }


```

✓ The code meets the specification.

Formal Verification Request 7

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

📅 31, Jul 2019

 25.08 ms

Line 114 in File BGBP.sol

114 `//@CTK NO_BUF_OVERFLOW`

Line 130-142 in File BGBP.sol


```
130     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool
        success) {
131         require(!isBlackListed[msg.sender], "can't transfer token from address in black
            list");
132         require(!isBlackListed[_to], "can't transfer token to address in black list");
133         if (deprecated) {
134             success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.
                sender, _to, _value);
135             require(success, "failed to call upgraded contract");
136             return true;
137         } else {
138             return super.transfer(_to, _value);
139         }
140     }
```

 The code meets the specification.

Formal Verification Request 8

Method will not encounter an assertion failure.

 31, Jul 2019

 27.73 ms

Line 115 in File BGBP.sol

115 `//@CTK NO_ASF`

Line 130-142 in File BGBP.sol


```
130     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool
        success) {
131         require(!isBlackListed[msg.sender], "can't transfer token from address in black
            list");
132         require(!isBlackListed[_to], "can't transfer token to address in black list");
133         if (deprecated) {
134             success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.
                sender, _to, _value);
135             require(success, "failed to call upgraded contract");
136             return true;
137         } else {
138             return super.transfer(_to, _value);
139         }
140     }
```

 The code meets the specification.

Formal Verification Request 9

transfer

 31, Jul 2019

 215.87 ms

Line 116-129 in File BGBP.sol

```
116  /*@CTK transfer
117     @tag assume_completion
118     @pre !deprecated
119     @post !_paused
120     @post !isBlackListed[msg.sender]
121     @post !isBlackListed[_to]
122     @post (_to) != (address(0))
123     @post (_value) <= (_balances[msg.sender])
124     @post (msg.sender != _to) -> (__post._balances[_to] == _balances[_to] + _value)
125     @post (msg.sender != _to) -> (__post._balances[msg.sender] == _balances[msg.
        sender] - _value)
126     @post (msg.sender == _to) -> (__post._balances[_to] == _balances[_to])
127     @post (msg.sender == _to) -> (__post._balances[msg.sender] == _balances[msg.
        sender])
128     @post success == true
129  */
```

Line 130-142 in File BGBP.sol


```
130  function transfer(address _to, uint256 _value) public whenNotPaused returns (bool
        success) {
131      require(!isBlackListed[msg.sender], "can't transfer token from address in black
        list");
132      require(!isBlackListed[_to], "can't transfer token to address in black list");
133      if (deprecated) {
134          success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.
            sender, _to, _value);
135          require(success, "failed to call upgraded contract");
136          return true;
137      } else {
138          return super.transfer(_to, _value);
139      }
140  }
```

 The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

 31, Jul 2019

 324.67 ms

Line 145 in File BGBP.sol

```
145  //@CTK NO_OVERFLOW
```

Line 163-175 in File BGBP.sol


```

163     function transferFrom(address _from, address _to, uint256 _value) public
164         whenNotPaused returns (bool success) {
165         require(!isBlackListed[_from], "can't transfer token from address in black list");
166         require(!isBlackListed[_to], "can't transfer token to address in black list");
167         if (deprecated) {
168             success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
169                 sender, _from, _to, _value);
170             require(success, "failed to call upgraded contract");
171             return true;
172         } else {
173             return super.transferFrom(_from, _to, _value);
174         }
175     }


```

✓ The code meets the specification.

Formal Verification Request 11

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

 31, Jul 2019

 59.87 ms

Line 146 in File BGBP.sol

```
146 // @CTK NO_BUF_OVERFLOW
```

Line 163-175 in File BGBP.sol

```

163     function transferFrom(address _from, address _to, uint256 _value) public
164         whenNotPaused returns (bool success) {
165         require(!isBlackListed[_from], "can't transfer token from address in black list");
166         require(!isBlackListed[_to], "can't transfer token to address in black list");
167         if (deprecated) {
168             success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
169                 sender, _from, _to, _value);
170             require(success, "failed to call upgraded contract");
171             return true;
172         } else {
173             return super.transferFrom(_from, _to, _value);
174         }
175     }


```

✓ The code meets the specification.

Formal Verification Request 12

Method will not encounter an assertion failure.

 31, Jul 2019

 60.64 ms

Line 147 in File BGBP.sol

147 //CTK NO_ASF

Line 163-175 in File BGBP.sol

```

163 function transferFrom(address _from, address _to, uint256 _value) public
    whenNotPaused returns (bool success) {
164     require(!isBlackListed[_from], "can't transfer token from address in black list
        ");
165     require(!isBlackListed[_to], "can't transfer token to address in black list");
166     if (deprecated) {
167         success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
            sender, _from, _to, _value);
168         require(success, "failed to call upgraded contract");
169         return true;
170     } else {
171         return super.transferFrom(_from, _to, _value);
172     }
173 }

```

✓ The code meets the specification.

Formal Verification Request 13

transferFrom

📅 31, Jul 2019

🕒 652.39 ms

Line 148-162 in File BGBP.sol

```

148 /*CTK "transferFrom"
149     @tag assume_completion
150     @pre !deprecated
151     @post !_paused
152     @post !isBlackListed[_from]
153     @post !isBlackListed[_to]
154     @post (_to) != (address(0))
155     @post (_value) <= (_balances[_from])
156     @post (_value) <= (_allowances[_from][msg.sender])
157     @post (_from != _to) -> (__post._balances[_to] == (_balances[_to] + _value))
158     @post (_from != _to) -> (__post._balances[_from] == (_balances[_from] - _value))
159     @post (_from == _to) -> (__post._balances[_to] == _balances[_to])
160     @post (_from == _to) -> (__post._balances[_from] == _balances[_from])
161     @post (__post._allowances[_from][msg.sender]) == (_allowances[_from][msg.sender]
        - _value)
162 */

```

Line 163-175 in File BGBP.sol

```

163 function transferFrom(address _from, address _to, uint256 _value) public
    whenNotPaused returns (bool success) {
164     require(!isBlackListed[_from], "can't transfer token from address in black list
        ");
165     require(!isBlackListed[_to], "can't transfer token to address in black list");
166     if (deprecated) {
167         success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
            sender, _from, _to, _value);
168         require(success, "failed to call upgraded contract");

```


```
169         return true;
170     } else {
171         return super.transferFrom(_from, _to, _value);
172     }
173 }
```

✓ The code meets the specification.

Formal Verification Request 14

balanceOf

 31, Jul 2019

 58.28 ms

Line 178-182 in File BGBP.sol

```
178  /*@CTK balanceOf
179     @tag assume_completion
180     @pre !deprecated
181     @post __return == _balances[who]
182  */
```

Line 183-189 in File BGBP.sol


```
183  function balanceOf(address who) public view returns (uint256) {
184      if (deprecated) {
185          return UpgradedStandardToken(upgradedAddress).balanceOf(who);
186      } else {
187          return super.balanceOf(who);
188      }
189  }
```

✓ The code meets the specification.

Formal Verification Request 15

approve

 31, Jul 2019

 102.98 ms

Line 192-199 in File BGBP.sol

```
192  /*@CTK approve
193     @tag assume_completion
194     @pre !deprecated
195     @pre !_paused
196     @pre _spender != 0x0
197     @pre _spender != msg.sender
198     @post (__post._allowances[msg.sender][_spender]) == (_value)
199  */
```

Line 200-210 in File BGBP.sol

```

200     function approve(address _spender, uint256 _value) public whenNotPaused returns (
201         bool success) {
202         if (deprecated) {
203             success = UpgradedStandardToken(upgradedAddress).approveByLegacy(msg.sender
204                 , _spender, _value);
205             require(success, "failed to call upgraded contract");
206             return true;
207         } else {
208             return super.approve(_spender, _value);
209         }
210     }


```

✓ The code meets the specification.

Formal Verification Request 16

increaseAllowance

 31, Jul 2019

 131.61 ms

Line 213-220 in File BGBP.sol

```

213     /*@CTK increaseAllowance
214         @tag assume_completion
215         @pre !deprecated
216         @pre !_paused
217         @pre _spender != 0x0
218         @pre _spender != msg.sender
219         @post (__post._allowances[msg.sender][_spender]) == (_allowances[msg.sender][
220             _spender] + _addedValue)
221     */

```

Line 221-231 in File BGBP.sol

```

221     function increaseAllowance(address _spender, uint256 _addedValue) public
222         whenNotPaused returns (bool success) {
223         if (deprecated) {
224             success = UpgradedStandardToken(upgradedAddress).increaseAllowanceByLegacy(
225                 msg.sender, _spender, _addedValue);
226             require(success, "failed to call upgraded contract");
227             return true;
228         } else {
229             return super.increaseAllowance(_spender, _addedValue);
230         }
231     }


```

✓ The code meets the specification.

Formal Verification Request 17

decreaseAllowance

 31, Jul 2019

 143.18 ms

Line 234-242 in File BGBP.sol

```

234  /*@CTK decreaseAllowance
235     @tag assume_completion
236     @pre !deprecated
237     @pre !_paused
238     @pre _spender != 0x0
239     @pre _spender != msg.sender
240     @post (_subtractedValue > _allowances[msg.sender][_spender]) -> (__post.
        _allowances[msg.sender][_spender] == 0)
241     @post (_subtractedValue <= _allowances[msg.sender][_spender]) -> (__post.
        _allowances[msg.sender][_spender] == _allowances[msg.sender][_spender] -
        _subtractedValue)
242  */

```

Line 243-253 in File BGBP.sol

```

243  function decreaseAllowance(address _spender, uint256 _subtractedValue) public
        whenNotPaused returns (bool success) {
244      if (deprecated) {
245          success = UpgradedStandardToken(upgradedAddress).decreaseAllowanceByLegacy(
                msg.sender, _spender, _subtractedValue);
246          require(success, "failed to call upgraded contract");
247          return true;
248      } else {
249          return super.decreaseAllowance(_spender, _subtractedValue);
250      }
251  }

```

✓ The code meets the specification.

Formal Verification Request 18

allowance



31, Jul 2019



52.38 ms

Line 256-260 in File BGBP.sol

```

256  /*@CTK allowance
257     @tag assume_completion
258     @pre !deprecated
259     @post remaining == _allowances[_owner][_spender]
260  */

```

Line 261-267 in File BGBP.sol

```

261  function allowance(address _owner, address _spender) public view returns (uint256
        remaining) {
262      if (deprecated) {
263          return UpgradedStandardToken(upgradedAddress).allowance(_owner, _spender);
264      } else {
265          return super.allowance(_owner, _spender);
266      }
267  }


```

✓ The code meets the specification.

Formal Verification Request 19

deprecate

 31, Jul 2019

 55.62 ms

Line 270-277 in File BGBP.sol

```
270  /*@CTK deprecate
271      @tag assume_completion
272      @post msg.sender == _owner
273      @post _upgradedAddress != address(0)
274      @post !deprecated
275      @post __post.deprecated == true
276      @post __post.upgradedAddress == _upgradedAddress
277  */
```

Line 278-285 in File BGBP.sol


```
278  function deprecate(address _upgradedAddress) public onlyOwner {
279      require(_upgradedAddress != address(0x0), "_upgradedAddress is a zero address")
280      ;
281      require(!deprecated, "this contract has been deprecated");
282      deprecated = true;
283      upgradedAddress = _upgradedAddress;
284      emit Deprecate(_upgradedAddress);
285  }
```

 The code meets the specification.

Formal Verification Request 20

totalSupply

 31, Jul 2019

 57.48 ms

Line 287-291 in File BGBP.sol

```
287  /*@CTK totalSupply
288      @tag assume_completion
289      @pre !deprecated
290      @post __return == _totalSupply
291  */
```

Line 292-298 in File BGBP.sol


```
292  function totalSupply() public view returns (uint256) {
293      if (deprecated) {
294          return UpgradedStandardToken(upgradedAddress).totalSupply();
295      } else {
296          return super.totalSupply();
297      }
298  }
```

 The code meets the specification.

Formal Verification Request 21

redeem

 31, Jul 2019

 228.59 ms

Line 304-311 in File BGBP.sol

```
304  /*@CTK redeem
305      @tag assume_completion
306      @post msg.sender == _owner
307      @post !_paused
308      @post deprecated == false
309      @post __post._totalSupply == _totalSupply + amount
310      @post __post._balances[msg.sender] == _balances[msg.sender] + amount
311  */
```

Line 312-317 in File BGBP.sol

```
312  function issue(uint256 amount) public onlyOwner whenNotPaused {
313      require(!deprecated, "this contract has been deprecated");
314
315      super._mint(msg.sender, amount);
316      emit Issue(amount);
317  }
```

 The code meets the specification.

Formal Verification Request 22

redeem

 31, Jul 2019

 300.04 ms

Line 324-331 in File BGBP.sol

```
324  /*@CTK redeem
325      @tag assume_completion
326      @post msg.sender == _owner
327      @post !_paused
328      @post deprecated == false
329      @post __post._totalSupply == _totalSupply - amount
330      @post __post._balances[msg.sender] == _balances[msg.sender] - amount
331  */
```

Line 332-337 in File BGBP.sol

```
332  function redeem(uint256 amount) public onlyOwner whenNotPaused {
333      require(!deprecated, "this contract has been deprecated");
334
335      super._burn(msg.sender, amount);
336      emit Redeem(amount);
337  }
```

 The code meets the specification.

Source Code with CertiK Labels

File BGBP.sol

```

1  pragma solidity 0.5.8;
2
3  import 'openzeppelin-solidity/contracts/token/ERC20/ERC20.sol';
4  import 'openzeppelin-solidity/contracts/lifecycle/Pausable.sol';
5  import 'openzeppelin-solidity/contracts/ownership/Ownable.sol';
6
7  contract BlackListableToken is Ownable, ERC20 {
8
9      ////////// Getters to allow the same blacklist to be used also by other contracts (
10         including upgraded Tether) //////////
11      /*@CTK getBlackListStatus
12         @tag assume_completion
13         @post __return == isBlackListed[_maker]
14      */
15      function getBlackListStatus(address _maker) external view returns (bool) {
16          return isBlackListed[_maker];
17      }
18
19      mapping (address => bool) public isBlackListed;
20
21      /*@CTK addBlackList
22         @tag assume_completion
23         @post msg.sender == _owner
24         @post isBlackListed[_evilUser] == false
25         @post __post.isBlackListed[_evilUser] == true
26      */
27      function addBlackList(address _evilUser) public onlyOwner {
28          require(!isBlackListed[_evilUser], "_evilUser is already in black list");
29
30          isBlackListed[_evilUser] = true;
31          emit AddedBlackList(_evilUser);
32      }
33
34      /*@CTK removeBlackList
35         @tag assume_completion
36         @post msg.sender == _owner
37         @post isBlackListed[_clearedUser] == true
38         @post __post.isBlackListed[_clearedUser] == false
39      */
40      function removeBlackList(address _clearedUser) public onlyOwner {
41          require(isBlackListed[_clearedUser], "_clearedUser isn't in black list");
42          isBlackListed[_clearedUser] = false;
43          emit RemovedBlackList(_clearedUser);
44      }
45
46      /*@CTK destroyBlackFunds
47         @tag assume_completion
48         @post msg.sender == _owner
49         @post _blackListedUser != address(0)
50         @post isBlackListed[_blackListedUser] == true
51         @post __post._balances[_blackListedUser] == 0
52         @post __post._totalSupply == _totalSupply - _balances[_blackListedUser]
53      */
54      function destroyBlackFunds(address _blackListedUser) public onlyOwner {

```



```

54     require(_blackListedUser != address(0x0), "_blackListedUser is the zero address
55         ");
56     require(isBlackListed[_blackListedUser], "_blackListedUser isn't in black list"
57         );
58     uint256 dirtyFunds = balanceOf(_blackListedUser);
59     super._burn(_blackListedUser, dirtyFunds);
60     emit DestroyedBlackFunds(_blackListedUser, dirtyFunds);
61 }
62 event DestroyedBlackFunds(address indexed _blackListedUser, uint256 _balance);
63
64 event AddedBlackList(address indexed _user);
65
66 event RemovedBlackList(address indexed _user);
67
68 }
69
70 contract UpgradedStandardToken is ERC20 {
71     // those methods are called by the legacy contract
72     // and they must ensure msg.sender to be the contract address
73     function transferByLegacy(address from, address to, uint256 value) public returns
74         (bool);
75     function transferFromByLegacy(address sender, address from, address to, uint256
76         value) public returns (bool);
77     function approveByLegacy(address owner, address spender, uint256 value) public
78         returns (bool);
79     function increaseAllowanceByLegacy(address owner, address spender, uint256
80         addedValue) public returns (bool);
81     function decreaseAllowanceByLegacy(address owner, address spender, uint256
82         subtractedValue) public returns (bool);
83 }
84
85 contract BGBPToken is ERC20, Pausable, BlackListableToken {
86     string public name;
87     string public symbol;
88     uint8 public decimals;
89     address public upgradedAddress;
90     bool public deprecated;
91
92     // The contract can be initialized with a number of tokens
93     // All the tokens are deposited to the owner address
94     //
95     // @param _balance Initial supply of the contract
96     // @param _name Token Name
97     // @param _symbol Token symbol
98     // @param _decimals Token decimals
99     /*@CTK BGBPToken
100     @tag assume_completion
101     @post __post.name == _name
102     @post __post.symbol == _symbol
103     @post __post.decimals == _decimals
104     @post __post.deprecated == false
105     @post __post._balances[msg.sender] == _balances[msg.sender] + _initialSupply
106     @post __post._totalSupply == _totalSupply + _initialSupply
107     */

```

```

104     constructor(uint256 _initialSupply, string memory _name, string memory _symbol,
105                 uint8 _decimals) public {
106         name = _name;
107         symbol = _symbol;
108         decimals = _decimals;
109         deprecated = false;
110         super._mint(msg.sender, _initialSupply);
111     }
112
113     // Forward ERC20 methods to upgraded contract if this one is deprecated
114     //@CTK NO_OVERFLOW
115     //@CTK NO_BUF_OVERFLOW
116     //@CTK NO_ASF
117     /*@CTK transfer
118     @tag assume_completion
119     @pre !deprecated
120     @post !_paused
121     @post !isBlackListed[msg.sender]
122     @post !isBlackListed[_to]
123     @post (_to) != (address(0))
124     @post (_value) <= (_balances[msg.sender])
125     @post (msg.sender != _to) -> (__post._balances[_to] == _balances[_to] + _value)
126     @post (msg.sender != _to) -> (__post._balances[msg.sender] == _balances[msg.
127         sender] - _value)
128     @post (msg.sender == _to) -> (__post._balances[_to] == _balances[_to])
129     @post (msg.sender == _to) -> (__post._balances[msg.sender] == _balances[msg.
130         sender])
131     @post success == true
132 */
133     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool
134         success) {
135         require(!isBlackListed[msg.sender], "can't transfer token from address in black
136             list");
137         require(!isBlackListed[_to], "can't transfer token to address in black list");
138         if (deprecated) {
139             success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.
140                 sender, _to, _value);
141             require(success, "failed to call upgraded contract");
142             return true;
143         } else {
144             return super.transfer(_to, _value);
145         }
146     }
147
148     // Forward ERC20 methods to upgraded contract if this one is deprecated
149     //@CTK NO_OVERFLOW
150     //@CTK NO_BUF_OVERFLOW
151     //@CTK NO_ASF
152     /*@CTK "transferFrom"
153     @tag assume_completion
154     @pre !deprecated
155     @post !_paused
156     @post !isBlackListed[_from]
157     @post !isBlackListed[_to]
158     @post (_to) != (address(0))
159     @post (_value) <= (_balances[_from])
160     @post (_value) <= (_allowances[_from][msg.sender])
161     @post (_from != _to) -> (__post._balances[_to] == (_balances[_to] + _value))

```

```

156     @post (_from != _to) -> (__post._balances[_from] == (_balances[_from] - _value))
157     @post (_from == _to) -> (__post._balances[_to] == _balances[_to])
158     @post (_from == _to) -> (__post._balances[_from] == _balances[_from])
159     @post (__post._allowances[_from][msg.sender]) == (_allowances[_from][msg.sender]
        - _value)
160 */
161 function transferFrom(address _from, address _to, uint256 _value) public
    whenNotPaused returns (bool success) {
162     require(!isBlackListed[_from], "can't transfer token from address in black list
        ");
163     require(!isBlackListed[_to], "can't transfer token to address in black list");
164     if (deprecated) {
165         success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
            sender, _from, _to, _value);
166         require(success, "failed to call upgraded contract");
167         return true;
168     } else {
169         return super.transferFrom(_from, _to, _value);
170     }
171 }
172
173 // Forward ERC20 methods to upgraded contract if this one is deprecated
174 /*@CTK balanceOf
175     @tag assume_completion
176     @pre !deprecated
177     @post __return == _balances[who]
178 */
179 function balanceOf(address who) public view returns (uint256) {
180     if (deprecated) {
181         return UpgradedStandardToken(upgradedAddress).balanceOf(who);
182     } else {
183         return super.balanceOf(who);
184     }
185 }
186
187 // Forward ERC20 methods to upgraded contract if this one is deprecated
188 /*@CTK approve
189     @tag assume_completion
190     @pre !deprecated
191     @pre !_paused
192     @pre _spender != 0x0
193     @pre _spender != msg.sender
194     @post (__post._allowances[msg.sender][_spender]) == (_value)
195 */
196 function approve(address _spender, uint256 _value) public whenNotPaused returns (
    bool success) {
197     if (deprecated) {
198         success = UpgradedStandardToken(upgradedAddress).approveByLegacy(msg.sender
            , _spender, _value);
199         require(success, "failed to call upgraded contract");
200         return true;
201     } else {
202         return super.approve(_spender, _value);
203     }
204 }
205
206 // Forward ERC20 methods to upgraded contract if this one is deprecated
207 /*@CTK increaseAllowance

```

```

208     @tag assume_completion
209     @pre !deprecated
210     @pre !_paused
211     @pre _spender != 0x0
212     @pre _spender != msg.sender
213     @post (__post._allowances[msg.sender][_spender]) == (_allowances[msg.sender][
        _spender] + _addedValue)
214 */
215 function increaseAllowance(address _spender, uint256 _addedValue) public
    whenNotPaused returns (bool success) {
216     if (deprecated) {
217         success = UpgradedStandardToken(upgradedAddress).increaseAllowanceByLegacy(
            msg.sender, _spender, _addedValue);
218         require(success, "failed to call upgraded contract");
219         return true;
220     } else {
221         return super.increaseAllowance(_spender, _addedValue);
222     }
223 }
224
225 // Forward ERC20 methods to upgraded contract if this one is deprecated
226 /*@CTK decreaseAllowance
227     @tag assume_completion
228     @pre !deprecated
229     @pre !_paused
230     @pre _spender != 0x0
231     @pre _spender != msg.sender
232     @post (_subtractedValue > _allowances[msg.sender][_spender]) -> (__post.
        _allowances[msg.sender][_spender] == 0)
233     @post (_subtractedValue <= _allowances[msg.sender][_spender]) -> (__post.
        _allowances[msg.sender][_spender] == _allowances[msg.sender][_spender] -
        _subtractedValue)
234 */
235 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
    whenNotPaused returns (bool success) {
236     if (deprecated) {
237         success = UpgradedStandardToken(upgradedAddress).decreaseAllowanceByLegacy(
            msg.sender, _spender, _subtractedValue);
238         require(success, "failed to call upgraded contract");
239         return true;
240     } else {
241         return super.decreaseAllowance(_spender, _subtractedValue);
242     }
243 }
244
245 // Forward ERC20 methods to upgraded contract if this one is deprecated
246 /*@CTK allowance
247     @tag assume_completion
248     @pre !deprecated
249     @post remaining == _allowances[_owner][_spender]
250 */
251 function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
252     if (deprecated) {
253         return UpgradedStandardToken(upgradedAddress).allowance(_owner, _spender);
254     } else {
255         return super.allowance(_owner, _spender);
256     }

```

```

257     }
258
259     // deprecate current contract in favour of a new one
260     /*@CTK deprecate
261         @tag assume_completion
262         @post msg.sender == _owner
263         @post _upgradedAddress != address(0)
264         @post !deprecated
265         @post __post.deprecated == true
266         @post __post.upgradedAddress == _upgradedAddress
267     */
268     function deprecate(address _upgradedAddress) public onlyOwner {
269         require(_upgradedAddress != address(0x0), "_upgradedAddress is a zero address")
270         ;
271         require(!deprecated, "this contract has been deprecated");
272
273         deprecated = true;
274         upgradedAddress = _upgradedAddress;
275         emit Deprecate(_upgradedAddress);
276     }
277
278     /*@CTK totalSupply
279         @tag assume_completion
280         @pre !deprecated
281         @post __return == _totalSupply
282     */
283     function totalSupply() public view returns (uint256) {
284         if (deprecated) {
285             return UpgradedStandardToken(upgradedAddress).totalSupply();
286         } else {
287             return super.totalSupply();
288         }
289     }
290
291     // Issue a new amount of tokens
292     // these tokens are deposited into the owner address
293     //
294     // @param _amount Number of tokens to be issued
295     /*@CTK redeem
296         @tag assume_completion
297         @post msg.sender == _owner
298         @post !_paused
299         @post deprecated == false
300         @post __post._totalSupply == _totalSupply + amount
301         @post __post._balances[msg.sender] == _balances[msg.sender] + amount
302     */
303     function issue(uint256 amount) public onlyOwner whenNotPaused {
304         require(!deprecated, "this contract has been deprecated");
305
306         super._mint(msg.sender, amount);
307         emit Issue(amount);
308     }
309
310     // Redeem tokens.
311     // These tokens are withdrawn from the owner address
312     // if the balance must be enough to cover the redeem
313     // or the call will fail.
314     // @param _amount Number of tokens to be issued

```

```

314  /*@CTK redeem
315      @tag assume_completion
316      @post msg.sender == _owner
317      @post !_paused
318      @post deprecated == false
319      @post __post._totalSupply == _totalSupply - amount
320      @post __post._balances[msg.sender] == _balances[msg.sender] - amount
321  */
322  function redeem(uint256 amount) public onlyOwner whenNotPaused {
323      require(!deprecated, "this contract has been deprecated");
324
325      super._burn(msg.sender, amount);
326      emit Redeem(amount);
327  }
328
329  // Called when new token are issued
330  event Issue(uint256 amount);
331
332  // Called when tokens are redeemed
333  event Redeem(uint256 amount);
334
335  // Called when contract is deprecated
336  event Deprecate(address indexed newAddress);
337  }

```