CertiK Audit Report For Binance(BGBP)

Request Date: 2019-07-09
Revision Date: 2019-07-31
Platform Name: Ethereum

Contents

Disclaimer

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/

Exective 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 vulnerabilies, 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% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

“tx.origin” for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **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 $\boxed{\text{CRITICAL}}$ $\boxed{\text{IMPORTANT}}$ $\boxed{\text{INFO}}$ $\boxed{\text{DISCUSSION}}$ (in decreasing significance level).

BGBP.sol previous version

- $\boxed{\text{IMPORTANT}}$ `SafeMath`, `Ownable`, `BasicToken`, `StandardToken`: Usage of locally-included outdated contracts/libraries.
  - (Binance - Resolved) Use latest `openzeppelin-solidity` library.
- $\boxed{\text{IMPORTANT}}$ `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 BlackListed. 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(_updradedAddress != 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

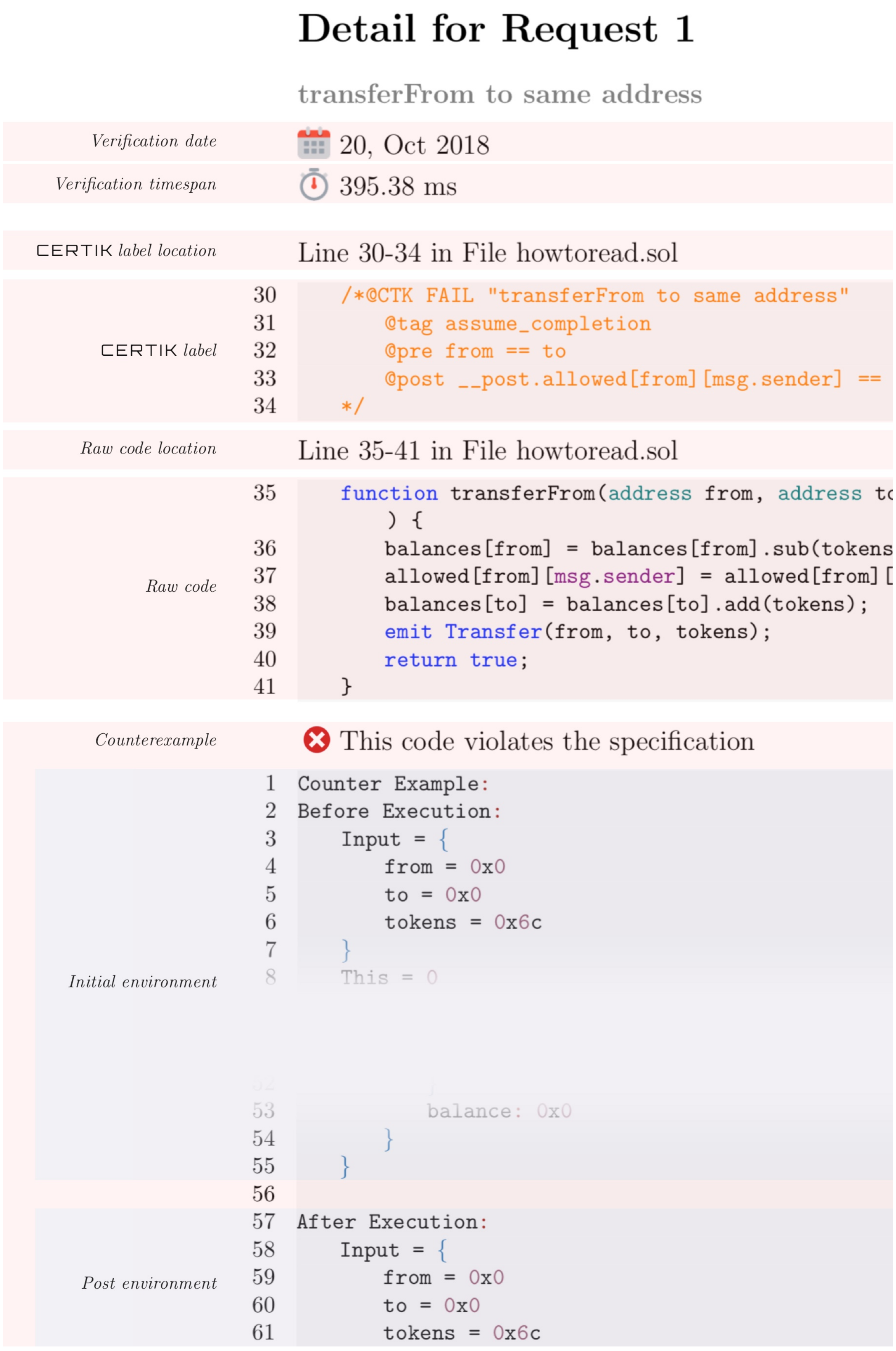
Line 1 in File BGBP.sol

```
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

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

                balance: 0x0
            }
    }
```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Formal Verification Request 1

getBlackListStatus

📅 31, Jul 2019
⏱ 8.99 ms

Line 10-13 in File BGBP.sol

```
    /*@CTK getBlackListStatus
      @tag assume_completion
      @post __return == isBlackListed[_maker]
     */
```

Line 14-16 in File BGBP.sol

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

✓ The code meets the specification.

Formal Verification Request 2

addBlackList

📅 31, Jul 2019
⏱ 63.15 ms

Line 20-25 in File BGBP.sol

```
    /*@CTK addBlackList
      @tag assume_completion
      @post msg.sender == _owner
      @post isBlackListed[_evilUser] == false
      @post __post.isBlackListed[_evilUser] == true
     */
```

Line 26-31 in File BGBP.sol

```
    function addBlackList(address _evilUser) public onlyOwner {
        require(!isBlackListed[_evilUser], "_evilUser is already in black list");

        isBlackListed[_evilUser] = true;
        emit AddedBlackList(_evilUser);
    }
```

✓ The code meets the specification.

Formal Verification Request 3

removeBlackList

📅 31, Jul 2019
⏱ 43.47 ms

Line 33-38 in File BGBP.sol

```
    /*@CTK removeBlackList
      @tag assume_completion
      @post msg.sender == _owner
      @post isBlackListed[_clearedUser] == true
      @post __post.isBlackListed[_clearedUser] == false
     */
```

Line 39-43 in File BGBP.sol

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

✓ The code meets the specification.

Formal Verification Request 4

destroyBlackFunds

📅 31, Jul 2019
⏱ 289.95 ms

Line 45-52 in File BGBP.sol

```
    /*@CTK destroyBlackFunds
      @tag assume_completion
      @post msg.sender == _owner
      @post _blackListedUser != address(0)
      @post isBlackListed[_blackListedUser] == true
      @post __post._balances[_blackListedUser] == 0
      @post __post._totalSupply == _totalSupply - _balances[_blackListedUser]
     */
```

Line 53-60 in File BGBP.sol

```
    function destroyBlackFunds(address _blackListedUser) public onlyOwner {
        require(_blackListedUser != address(0x0), "_blackListedUser is the zero address
            ");
        require(isBlackListed[_blackListedUser], "_blackListedUser isn't in black list"
            );

        uint256 dirtyFunds = balanceOf(_blackListedUser);
        super._burn(_blackListedUser, dirtyFunds);
        emit DestroyedBlackFunds(_blackListedUser, dirtyFunds);
    }
```

✓ The code meets the specification.

Formal Verification Request 5

BGBPToken

📅 31, Jul 2019
⏱ 199.54 ms

Line 95-103 in File BGBP.sol

```
/*@CTK BGBPToken
  @tag assume_completion
  @post __post.name == _name
  @post __post.symbol == _symbol
  @post __post.decimals == _decimals
  @post __post.deprecated == false
  @post __post._balances[msg.sender] == _balances[msg.sender] + _initialSupply
  @post __post._totalSupply == _totalSupply + _initialSupply
 */
```

Line 104-110 in File BGBP.sol

```
constructor(uint256 _initialSupply, string memory _name, string memory _symbol,
    uint8 _decimals) public {
    name = _name;
    symbol = _symbol;
    decimals = _decimals;
    deprecated = false;
    super._mint(msg.sender, _initialSupply);
}
```

✓ 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

```
//@CTK NO_OVERFLOW
```

Line 130-142 in File BGBP.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 130-142 in File BGBP.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 130-142 in File BGBP.sol

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

✓ The code meets the specification.

Formal Verification Request 9

transfer

📅 31, Jul 2019
⏱ 215.87 ms

Line 116-129 in File BGBP.sol

```
    /*@CTK transfer
      @tag assume_completion
      @pre !deprecated
      @post !_paused
      @post !isBlackListed[msg.sender]
      @post !isBlackListed[_to]
      @post (_to) != (address(0))
      @post (_value) <= (_balances[msg.sender])
      @post (msg.sender != _to) -> (__post._balances[_to] == _balances[_to] + _value)
      @post (msg.sender != _to) -> (__post._balances[msg.sender] == _balances[msg.
          sender] - _value)
      @post (msg.sender == _to) -> (__post._balances[_to] == _balances[_to])
      @post (msg.sender == _to) -> (__post._balances[msg.sender] == _balances[msg.
          sender])
      @post success == true
    */
```

Line 130-142 in File BGBP.sol

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

✅ 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

```
    //@CTK NO_OVERFLOW
```

Line 163-175 in File BGBP.sol

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

✓ 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

```
//@CTK NO_BUF_OVERFLOW
```

Line 163-175 in File BGBP.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 163-175 in File BGBP.sol

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

✓ The code meets the specification.

Formal Verification Request 13

transferFrom

31, Jul 2019
652.39 ms

Line 148-162 in File BGBP.sol

```
    /*@CTK "transferFrom"
      @tag assume_completion
      @pre !deprecated
      @post !_paused
      @post !isBlackListed[_from]
      @post !isBlackListed[_to]
      @post (_to) != (address(0))
      @post (_value) <= (_balances[_from])
      @post (_value) <= (_allowances[_from][msg.sender])
      @post (_from != _to) -> (__post._balances[_to] == (_balances[_to] + _value))
      @post (_from != _to) -> (__post._balances[_from] == (_balances[_from] - _value))
      @post (_from == _to) -> (__post._balances[_to] == _balances[_to])
      @post (_from == _to) -> (__post._balances[_from] == _balances[_from])
      @post (__post._allowances[_from][msg.sender]) == (_allowances[_from][msg.sender]
          - _value)
    */
```

Line 163-175 in File BGBP.sol

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

✓ The code meets the specification.

Formal Verification Request 14

balanceOf

31, Jul 2019
58.28 ms

Line 178-182 in File BGBP.sol

```
    /*@CTK balanceOf
      @tag assume_completion
      @pre !deprecated
      @post __return == _balances[who]
     */
```

Line 183-189 in File BGBP.sol

```
    function balanceOf(address who) public view returns (uint256) {
        if (deprecated) {
            return UpgradedStandardToken(upgradedAddress).balanceOf(who);
        } else {
            return super.balanceOf(who);
        }
    }
```

✓ The code meets the specification.

Formal Verification Request 15

approve

31, Jul 2019
102.98 ms

Line 192-199 in File BGBP.sol

```
    /*@CTK approve
      @tag assume_completion
      @pre !deprecated
      @pre !_paused
      @pre _spender != 0x0
      @pre _spender != msg.sender
      @post (__post._allowances[msg.sender][_spender]) == (_value)
    */
```

Line 200-210 in File BGBP.sol

```
    function approve(address _spender, uint256 _value) public whenNotPaused returns (
        bool success) {
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).approveByLegacy(msg.sender
                , _spender, _value);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.approve(_spender, _value);
        }
    }
```

✅ The code meets the specification.

Formal Verification Request 16

increaseAllowance

📅 31, Jul 2019
⏱ 131.61 ms

Line 213-220 in File BGBP.sol

```
    /*@CTK increaseAllowance
      @tag assume_completion
      @pre !deprecated
      @pre !_paused
      @pre _spender != 0x0
      @pre _spender != msg.sender
      @post (__post._allowances[msg.sender][_spender]) == (_allowances[msg.sender][
          _spender] + _addedValue)
    */
```

Line 221-231 in File BGBP.sol

```
    function increaseAllowance(address _spender, uint256 _addedValue) public
        whenNotPaused returns (bool success) {
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).increaseAllowanceByLegacy(
                msg.sender, _spender, _addedValue);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.increaseAllowance(_spender, _addedValue);
        }
    }
```

✅ The code meets the specification.

Formal Verification Request 17

decreaseAllowance

📅 31, Jul 2019
⏱ 143.18 ms

Line 234-242 in File BGBP.sol

```
/*@CTK decreaseAllowance
  @tag assume_completion
  @pre !deprecated
  @pre !_paused
  @pre _spender != 0x0
  @pre _spender != msg.sender
  @post (_subtractedValue > _allowances[msg.sender][_spender]) -> (__post.
      _allowances[msg.sender][_spender] == 0)
  @post (_subtractedValue <= _allowances[msg.sender][_spender]) -> (__post.
      _allowances[msg.sender][_spender] == _allowances[msg.sender][_spender] -
      _subtractedValue)
*/
```

Line 243-253 in File BGBP.sol

```
function decreaseAllowance(address _spender, uint256 _subtractedValue) public
    whenNotPaused returns (bool success) {
    if (deprecated) {
        success = UpgradedStandardToken(upgradedAddress).decreaseAllowanceByLegacy(
            msg.sender, _spender, _subtractedValue);
        require(success, "failed to call upgraded contract");
        return true;
    } else {
        return super.decreaseAllowance(_spender, _subtractedValue);
    }
}
```

✓ The code meets the specification.

Formal Verification Request 18

allowance

31, Jul 2019
52.38 ms

Line 256-260 in File BGBP.sol

```
/*@CTK allowance
  @tag assume_completion
  @pre !deprecated
  @post remaining == _allowances[_owner][_spender]
 */
```

Line 261-267 in File BGBP.sol

```
function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
    if (deprecated) {
        return UpgradedStandardToken(upgradedAddress).allowance(_owner, _spender);
    } else {
        return super.allowance(_owner, _spender);
    }
}
```

✓ The code meets the specification.

Formal Verification Request 19

deprecate

📅 31, Jul 2019
⏱ 55.62 ms

Line 270-277 in File BGBP.sol

```
    /*@CTK deprecate
      @tag assume_completion
      @post msg.sender == _owner
      @post _upgradedAddress != address(0)
      @post !deprecated
      @post __post.deprecated == true
      @post __post.upgradedAddress == _upgradedAddress
     */
```

Line 278-285 in File BGBP.sol

```
    function deprecate(address _upgradedAddress) public onlyOwner {
        require(_upgradedAddress != address(0x0), "_upgradedAddress is a zero address")
            ;
        require(!deprecated, "this contract has been deprecated");

        deprecated = true;
        upgradedAddress = _upgradedAddress;
        emit Deprecate(_upgradedAddress);
    }
```

✓ The code meets the specification.

Formal Verification Request 20

totalSupply

📅 31, Jul 2019
⏱ 57.48 ms

Line 287-291 in File BGBP.sol

```
    /*@CTK totalSupply
      @tag assume_completion
      @pre !deprecated
      @post __return == _totalSupply
     */
```

Line 292-298 in File BGBP.sol

```
    function totalSupply() public view returns (uint256) {
        if (deprecated) {
            return UpgradedStandardToken(upgradedAddress).totalSupply();
        } else {
            return super.totalSupply();
        }
    }
```

✓ The code meets the specification.

Formal Verification Request 21

redeem

31, Jul 2019
228.59 ms

Line 304-311 in File BGBP.sol

```
    /*@CTK redeem
      @tag assume_completion
      @post msg.sender == _owner
      @post !_paused
      @post deprecated == false
      @post __post._totalSupply == _totalSupply + amount
      @post __post._balances[msg.sender] == _balances[msg.sender] + amount
     */
```

Line 312-317 in File BGBP.sol

```
    function issue(uint256 amount) public onlyOwner whenNotPaused {
        require(!deprecated, "this contract has been deprecated");

        super._mint(msg.sender, amount);
        emit Issue(amount);
    }
```

✓ The code meets the specification.

Formal Verification Request 22

redeem

31, Jul 2019
300.04 ms

Line 324-331 in File BGBP.sol

```
    /*@CTK redeem
      @tag assume_completion
      @post msg.sender == _owner
      @post !_paused
      @post deprecated == false
      @post __post._totalSupply == _totalSupply - amount
      @post __post._balances[msg.sender] == _balances[msg.sender] - amount
     */
```

Line 332-337 in File BGBP.sol

```
    function redeem(uint256 amount) public onlyOwner whenNotPaused {
        require(!deprecated, "this contract has been deprecated");

        super._burn(msg.sender, amount);
        emit Redeem(amount);
    }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File BGBP.sol

```
pragma solidity 0.5.8;

import 'openzeppelin-solidity/contracts/token/ERC20/ERC20.sol';
import 'openzeppelin-solidity/contracts/lifecycle/Pausable.sol';
import 'openzeppelin-solidity/contracts/ownership/Ownable.sol';

contract BlackListableToken is Ownable, ERC20 {

    /////// Getters to allow the same blacklist to be used also by other contracts (
        including upgraded Tether) ///////
    /*@CTK getBlackListStatus
      @tag assume_completion
      @post __return == isBlackListed[_maker]
     */
    function getBlackListStatus(address _maker) external view returns (bool) {
        return isBlackListed[_maker];
    }

    mapping (address => bool) public isBlackListed;

    /*@CTK addBlackList
      @tag assume_completion
      @post msg.sender == _owner
      @post isBlackListed[_evilUser] == false
      @post __post.isBlackListed[_evilUser] == true
     */
    function addBlackList(address _evilUser) public onlyOwner {
        require(!isBlackListed[_evilUser], "_evilUser is already in black list");

        isBlackListed[_evilUser] = true;
        emit AddedBlackList(_evilUser);
    }

    /*@CTK removeBlackList
      @tag assume_completion
      @post msg.sender == _owner
      @post isBlackListed[_clearedUser] == true
      @post __post.isBlackListed[_clearedUser] == false
     */
    function removeBlackList(address _clearedUser) public onlyOwner {
        require(isBlackListed[_clearedUser], "_clearedUser isn't in black list");
        isBlackListed[_clearedUser] = false;
        emit RemovedBlackList(_clearedUser);
    }

    /*@CTK destroyBlackFunds
      @tag assume_completion
      @post msg.sender == _owner
      @post _blackListedUser != address(0)
      @post isBlackListed[_blackListedUser] == true
      @post __post._balances[_blackListedUser] == 0
      @post __post._totalSupply == _totalSupply - _balances[_blackListedUser]
     */
    function destroyBlackFunds(address _blackListedUser) public onlyOwner {
        require(_blackListedUser != address(0x0), "_blackListedUser is the zero address");
        require(isBlackListed[_blackListedUser], "_blackListedUser isn't in black list");

        uint256 dirtyFunds = balanceOf(_blackListedUser);
        super._burn(_blackListedUser, dirtyFunds);
        emit DestroyedBlackFunds(_blackListedUser, dirtyFunds);
    }

    event DestroyedBlackFunds(address indexed _blackListedUser, uint256 _balance);

    event AddedBlackList(address indexed _user);

    event RemovedBlackList(address indexed _user);

}

contract UpgradedStandardToken is ERC20 {
    // those methods are called by the legacy contract
    // and they must ensure msg.sender to be the contract address
    function transferByLegacy(address from, address to, uint256 value) public returns (bool);
    function transferFromByLegacy(address sender, address from, address to, uint256 value) public returns (bool);
    function approveByLegacy(address owner, address spender, uint256 value) public returns (bool);
    function increaseAllowanceByLegacy(address owner, address spender, uint256 addedValue) public returns (bool);
    function decreaseAllowanceByLegacy(address owner, address spender, uint256 subtractedValue) public returns (bool);
}

contract BGBPToken is ERC20, Pausable, BlackListableToken {

    string public name;
    string public symbol;
    uint8 public decimals;
    address public upgradedAddress;
    bool public deprecated;

    //  The contract can be initialized with a number of tokens
    //  All the tokens are deposited to the owner address
    //
    // @param _balance Initial supply of the contract
    // @param _name Token Name
    // @param _symbol Token symbol
    // @param _decimals Token decimals
    /*@CTK BGBPToken
      @tag assume_completion
      @post __post.name == _name
      @post __post.symbol == _symbol
      @post __post.decimals == _decimals
      @post __post.deprecated == false
      @post __post._balances[msg.sender] == _balances[msg.sender] + _initialSupply
      @post __post._totalSupply == _totalSupply + _initialSupply
     */
```

```
constructor(uint256 _initialSupply, string memory _name, string memory _symbol, uint8 _decimals) public {
    name = _name;
    symbol = _symbol;
    decimals = _decimals;
    deprecated = false;
    super._mint(msg.sender, _initialSupply);
}

// Forward ERC20 methods to upgraded contract if this one is deprecated
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK transfer
  @tag assume_completion
  @pre !deprecated
  @post !_paused
  @post !isBlackListed[msg.sender]
  @post !isBlackListed[_to]
  @post (_to) != (address(0))
  @post (_value) <= (_balances[msg.sender])
  @post (msg.sender != _to) -> (__post._balances[_to] == _balances[_to] + _value)
  @post (msg.sender != _to) -> (__post._balances[msg.sender] == _balances[msg.sender] - _value)
  @post (msg.sender == _to) -> (__post._balances[_to] == _balances[_to])
  @post (msg.sender == _to) -> (__post._balances[msg.sender] == _balances[msg.sender])
  @post success == true
*/
function transfer(address _to, uint256 _value) public whenNotPaused returns (bool success) {
    require(!isBlackListed[msg.sender], "can't transfer token from address in black list");
    require(!isBlackListed[_to], "can't transfer token to address in black list");
    if (deprecated) {
        success = UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.sender, _to, _value);
        require(success, "failed to call upgraded contract");
        return true;
    } else {
        return super.transfer(_to, _value);
    }
}

// Forward ERC20 methods to upgraded contract if this one is deprecated
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "transferFrom"
  @tag assume_completion
  @pre !deprecated
  @post !_paused
  @post !isBlackListed[_from]
  @post !isBlackListed[_to]
  @post (_to) != (address(0))
  @post (_value) <= (_balances[_from])
  @post (_value) <= (_allowances[_from][msg.sender])
  @post (_from != _to) -> (__post._balances[_to] == (_balances[_to] + _value))
```

```
      @post (_from != _to) -> (__post._balances[_from] == (_balances[_from] - _value))
      @post (_from == _to) -> (__post._balances[_to] == _balances[_to])
      @post (_from == _to) -> (__post._balances[_from] == _balances[_from])
      @post (__post._allowances[_from][msg.sender]) == (_allowances[_from][msg.sender]
           - _value)
    */
    function transferFrom(address _from, address _to, uint256 _value) public
        whenNotPaused returns (bool success) {
        require(!isBlackListed[_from], "can't transfer token from address in black list
            ");
        require(!isBlackListed[_to], "can't transfer token to address in black list");
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).transferFromByLegacy(msg.
                sender, _from, _to, _value);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.transferFrom(_from, _to, _value);
        }
    }

    // Forward ERC20 methods to upgraded contract if this one is deprecated
    /*@CTK balanceOf
      @tag assume_completion
      @pre !deprecated
      @post __return == _balances[who]
     */
    function balanceOf(address who) public view returns (uint256) {
        if (deprecated) {
            return UpgradedStandardToken(upgradedAddress).balanceOf(who);
        } else {
            return super.balanceOf(who);
        }
    }

    // Forward ERC20 methods to upgraded contract if this one is deprecated
    /*@CTK approve
      @tag assume_completion
      @pre !deprecated
      @pre !_paused
      @pre _spender != 0x0
      @pre _spender != msg.sender
      @post (__post._allowances[msg.sender][_spender]) == (_value)
    */
    function approve(address _spender, uint256 _value) public whenNotPaused returns (
        bool success) {
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).approveByLegacy(msg.sender
                , _spender, _value);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.approve(_spender, _value);
        }
    }

    // Forward ERC20 methods to upgraded contract if this one is deprecated
    /*@CTK increaseAllowance
```

```
      @tag assume_completion
      @pre !deprecated
      @pre !_paused
      @pre _spender != 0x0
      @pre _spender != msg.sender
      @post (__post._allowances[msg.sender][_spender]) == (_allowances[msg.sender][
          _spender] + _addedValue)
    */
    function increaseAllowance(address _spender, uint256 _addedValue) public
        whenNotPaused returns (bool success) {
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).increaseAllowanceByLegacy(
                msg.sender, _spender, _addedValue);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.increaseAllowance(_spender, _addedValue);
        }
    }

    // Forward ERC20 methods to upgraded contract if this one is deprecated
    /*@CTK decreaseAllowance
      @tag assume_completion
      @pre !deprecated
      @pre !_paused
      @pre _spender != 0x0
      @pre _spender != msg.sender
      @post (_subtractedValue > _allowances[msg.sender][_spender]) -> (__post.
          _allowances[msg.sender][_spender] == 0)
      @post (_subtractedValue <= _allowances[msg.sender][_spender]) -> (__post.
          _allowances[msg.sender][_spender] == _allowances[msg.sender][_spender] -
          _subtractedValue)
    */
    function decreaseAllowance(address _spender, uint256 _subtractedValue) public
        whenNotPaused returns (bool success) {
        if (deprecated) {
            success = UpgradedStandardToken(upgradedAddress).decreaseAllowanceByLegacy(
                msg.sender, _spender, _subtractedValue);
            require(success, "failed to call upgraded contract");
            return true;
        } else {
            return super.decreaseAllowance(_spender, _subtractedValue);
        }
    }

    // Forward ERC20 methods to upgraded contract if this one is deprecated
    /*@CTK allowance
      @tag assume_completion
      @pre !deprecated
      @post remaining == _allowances[_owner][_spender]
     */
    function allowance(address _owner, address _spender) public view returns (uint256
        remaining) {
        if (deprecated) {
            return UpgradedStandardToken(upgradedAddress).allowance(_owner, _spender);
        } else {
            return super.allowance(_owner, _spender);
        }
    }

    // deprecate current contract in favour of a new one
    /*@CTK deprecate
      @tag assume_completion
      @post msg.sender == _owner
      @post _upgradedAddress != address(0)
      @post !deprecated
      @post __post.deprecated == true
      @post __post.upgradedAddress == _upgradedAddress
     */
    function deprecate(address _upgradedAddress) public onlyOwner {
        require(_upgradedAddress != address(0x0), "_upgradedAddress is a zero address")
            ;
        require(!deprecated, "this contract has been deprecated");

        deprecated = true;
        upgradedAddress = _upgradedAddress;
        emit Deprecate(_upgradedAddress);
    }

    /*@CTK totalSupply
      @tag assume_completion
      @pre !deprecated
      @post __return == _totalSupply
     */
    function totalSupply() public view returns (uint256) {
        if (deprecated) {
            return UpgradedStandardToken(upgradedAddress).totalSupply();
        } else {
            return super.totalSupply();
        }
    }

    // Issue a new amount of tokens
    // these tokens are deposited into the owner address
    //
    // @param _amount Number of tokens to be issued
    /*@CTK redeem
      @tag assume_completion
      @post msg.sender == _owner
      @post !_paused
      @post deprecated == false
      @post __post._totalSupply == _totalSupply + amount
      @post __post._balances[msg.sender] == _balances[msg.sender] + amount
     */
    function issue(uint256 amount) public onlyOwner whenNotPaused {
        require(!deprecated, "this contract has been deprecated");

        super._mint(msg.sender, amount);
        emit Issue(amount);
    }

    // Redeem tokens.
    // These tokens are withdrawn from the owner address
    // if the balance must be enough to cover the redeem
    // or the call will fail.
    // @param _amount Number of tokens to be issued
    /*@CTK redeem
      @tag assume_completion
      @post msg.sender == _owner
      @post !_paused
      @post deprecated == false
      @post __post._totalSupply == _totalSupply - amount
      @post __post._balances[msg.sender] == _balances[msg.sender] - amount
     */
    function redeem(uint256 amount) public onlyOwner whenNotPaused {
        require(!deprecated, "this contract has been deprecated");

        super._burn(msg.sender, amount);
        emit Redeem(amount);
    }

    // Called when new token are issued
    event Issue(uint256 amount);

    // Called when tokens are redeemed
    event Redeem(uint256 amount);

    // Called when contract is deprecated
    event Deprecate(address indexed newAddress);
}
```