Audit Report

Produced by CertiK for Micro Tuber, Smart Farm Technology

December 8, 2020
CERTIK AUDIT REPORT
FOR MICRO TUBER

Request Date: 2020-12-07
Revision Date: 2020-12-08
Platform Name: Ethereum
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disclaimer</td>
<td>1</td>
</tr>
<tr>
<td>About CertiK</td>
<td>2</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>3</td>
</tr>
<tr>
<td>Vulnerability Classification</td>
<td>3</td>
</tr>
<tr>
<td>Testing Summary</td>
<td>4</td>
</tr>
<tr>
<td>Audit Score</td>
<td>4</td>
</tr>
<tr>
<td>Type of Issues</td>
<td>4</td>
</tr>
<tr>
<td>Vulnerability Details</td>
<td>5</td>
</tr>
<tr>
<td>Review Notes</td>
<td>6</td>
</tr>
<tr>
<td>Static Analysis Results</td>
<td>7</td>
</tr>
<tr>
<td>Formal Verification Results</td>
<td>8</td>
</tr>
<tr>
<td>How to read</td>
<td>8</td>
</tr>
<tr>
<td>Source Code with CertiK Labels</td>
<td>31</td>
</tr>
</tbody>
</table>
Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.

- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.

- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.
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, CertiK’s mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance’s BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.
Executive Summary

This report has been prepared for Micro Tuber to discover issues and vulnerabilities in the source code of their MicroTuber smart contracts. A comprehensive examination has been performed, utilizing CertiK’s Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices 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

CertiK categorizes issues into three buckets based on overall risk levels:

**Critical**

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

**Medium**

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

**Low**

Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.
## Type of Issues

CertiK’s smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Issues</th>
<th>SWC ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer Overflow/Underflow</td>
<td>An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.</td>
<td>0</td>
<td>SWC-101</td>
</tr>
<tr>
<td>Function Incorrectness</td>
<td>Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Buffer Overflow</td>
<td>An attacker can write to arbitrary storage locations of a contract if array of out bound happens</td>
<td>0</td>
<td>SWC-124</td>
</tr>
<tr>
<td>Reentrancy</td>
<td>A malicious contract can call back into the calling contract before the first invocation of the function is finished.</td>
<td>0</td>
<td>SWC-107</td>
</tr>
<tr>
<td>Transaction Order Dependence</td>
<td>A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.</td>
<td>0</td>
<td>SWC-114</td>
</tr>
<tr>
<td>Timestamp Dependence</td>
<td>Timestamp can be influenced by miners to some degree.</td>
<td>0</td>
<td>SWC-116</td>
</tr>
<tr>
<td>Insecure Compiler Version</td>
<td>Using a 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.</td>
<td>0</td>
<td>SWC-102, SWC-103</td>
</tr>
<tr>
<td>Insecure Randomness</td>
<td>Using block attributes to generate random numbers is unreliable, as they can be influenced by miners to some degree.</td>
<td>0</td>
<td>SWC-120</td>
</tr>
<tr>
<td>“tx.origin” for Authorization</td>
<td>tx.origin should not be used for authorization. Use msg.sender instead.</td>
<td>0</td>
<td>SWC-115</td>
</tr>
<tr>
<td>Title</td>
<td>Description</td>
<td>Issues</td>
<td>SWC ID</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Delegatecall to Untrusted Callee</td>
<td>Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.</td>
<td>0</td>
<td>SWC-112</td>
</tr>
<tr>
<td>State Variable Default Visibility</td>
<td>Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.</td>
<td>0</td>
<td>SWC-108</td>
</tr>
<tr>
<td>Function Default Visibility</td>
<td>Functions are public by default, meaning a malicious user can make unauthorized or unintended state changes if a developer forgot to set the visibility.</td>
<td>0</td>
<td>SWC-100</td>
</tr>
<tr>
<td>Uninitialized Variables</td>
<td>Uninitialized local storage variables can point to other unexpected storage variables in the contract.</td>
<td>0</td>
<td>SWC-109</td>
</tr>
<tr>
<td>Assertion Failure</td>
<td>The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.</td>
<td>0</td>
<td>SWC-110</td>
</tr>
<tr>
<td>Deprecated Solidity Features</td>
<td>Several functions and operators in Solidity are deprecated and should not be used.</td>
<td>0</td>
<td>SWC-111</td>
</tr>
<tr>
<td>Unused Variables</td>
<td>Unused variables reduce code quality</td>
<td>0</td>
<td>SWC-131</td>
</tr>
</tbody>
</table>

**Vulnerability Details**

**Critical**

No issue found.

**Medium**

No issue found.

**Low**

No issue found.
Review Notes

Source Code SHA-256 Checksum

- ERC20PresetMinterPauser.sol
  301f2a6e734b8aa39373a41f36f5cdc5eaba61c83ce4cb89676245aa8f2b3b52
- source code

Summary

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

Meanwhile, it is recommended to have a more well-detailed document for the public to describe the source code specifications and implementations.

Overall we found the erc20 contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrive controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is not vulnerable to any classically known antipatterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend seeking multiple opinions, more test coverage and sandbox deployments before the mainnet release.

Recommendations

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

ERC20PresetMinterPauser.sol

- INFO there are duplicate code like // SPDX-License-Identifier: MIT,pragma solidity >=0.6.0 <0.8.0; which will result in compile failure. Besides, this file contains too much code. Recommend splitting this file into serveral files based on their function.
Static Analysis Results

INSECURE_COMPILER_VERSION
Line 3 in File ERC20PresetMinterPauser.sol

```solidity
pragma solidity >=0.6.0 <0.8.0;
```

ℹ️ Only these compiler versions are safe to compile your code: 0.7.4

INSECURE_COMPILER_VERSION
Line 3 in File ERC20PresetMinterPauser_Context.sol

```solidity
pragma solidity >=0.6.0 <0.8.0;
```

ℹ️ Only these compiler versions are safe to compile your code: 0.7.4
Formal Verification Results

How to read

**Detail for Request 1**

transferFrom to same address

<table>
<thead>
<tr>
<th>Verification date</th>
<th>20, Oct 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification timespan</td>
<td>395.38 ms</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CERTIK label</th>
<th>Line 30-34 in File howtoread.sol</th>
</tr>
</thead>
</table>
| 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

<table>
<thead>
<tr>
<th>Raw code</th>
<th>function transferFrom(address from, address to</th>
</tr>
</thead>
</table>
| 35 |)
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

1 Counter Example:
2 Before Execution:
3 Input = {
4 from = 0x0
5 to = 0x0
6 tokens = 0x6c
7 }
8 This = 0

53 balance: 0x0
54 }
55 }
56
57 After Execution:
58 Input = {
59 from = 0x0
60 to = 0x0
61 tokens = 0x6c
Formal Verification Request 1

SafeMath add

08, Dec 2020
26.84 ms

Line 34-41 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "SafeMath add"
@tag assume_completion
@post (a + b < a || a + b < b) == __reverted
@post !__reverted -> __return == a + b
@post !__reverted -> !__has_overflow
@post !__reverted -> !__has_assertion_failure
@post !(__has_buf_overflow)
*/

function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");

    return c;
}
```

The code meets the specification.

Formal Verification Request 2

SafeMath sub

08, Dec 2020
22.64 ms

Line 73-80 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "SafeMath sub"
@tag assume_completion
@post (a < b) == __reverted
@post !__reverted -> __return == a - b
@post !__reverted -> !__has_overflow
@post !__reverted -> !__has_assertion_failure
@post !(__has_buf_overflow)
*/
```

Line 81-86 in File ERC20PresetMinterPauser.sol

```solidity
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a - b;
    require(c <= a, "SafeMath: subtraction underflow");

    return c;
}
```
function sub(uint256 a, uint256 b, string memory errorMessage) internal

pure returns (uint256) {
    require(b <= a, errorMessage);
    uint256 c = a - b;
    return c;
}

The code meets the specification.

Formal Verification Request 3

SafeMath mul

08, Dec 2020
272.85 ms

Line 98-105 in File ERC20PresetMinterPauser.sol

/*@CTK "SafeMath mul"
@tag assume_completion
@post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (_, reverted)
@post !_reverted -> _return == a * b
@post !_reverted == !_has_overflow
@post !_reverted -> !_has_assertion_failure
@post !(_has_buf_overflow)
*/

Line 106-118 in File ERC20PresetMinterPauser.sol

function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See:
    // https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
    if (a == 0) {
        return 0;
    }
    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");
    return c;
}

The code meets the specification.
Formal Verification Request 4

SafeMath div

08, Dec 2020
19.34 ms

Line 148-155 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "SafeMath div"
@tag assume_completion
@post (b <= 0) == __reverted
@post !__reverted -> __return == a / b
@post !__reverted -> !__has_overflow
@post !__reverted -> !__has_assertion_failure
@post !(__has_buf_overflow)
*/
```

Line 156-162 in File ERC20PresetMinterPauser.sol

```solidity
function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
    require(b > 0, errorMessage);
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn’t hold
    return c;
}
```

The code meets the specification.

Formal Verification Request 5

SafeMath mod

08, Dec 2020
19.91 ms

Line 192-199 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "SafeMath mod"
@tag assume_completion
@post b != 0 -> !__reverted
@post !__reverted -> __return == a % b
@post !__reverted -> !__has_overflow
@post !(__has_buf_overflow)
@post !(__has_assertion_failure)
*/
```
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

---

**Formal Verification Request 6**

**ERC20 name**

08, Dec 2020

9.3 ms

---

**Formal Verification Request 7**

**ERC20 symbol**

08, Dec 2020

7.01 ms

---

---

---
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

function symbol() public view returns (string memory) {
    return _symbol;
}

The code meets the specification.

Formal Verification Request 8
ERC20 decimals

08, Dec 2020
8.93 ms

Line 369-372 in File ERC20PresetMinterPauser.sol

/*@CTK "ERC20 decimals"
@tag assume_completion
@post __return == __post._decimals */

function decimals() public view returns (uint8) {
    return _decimals;
}

The code meets the specification.

Formal Verification Request 9
ERC20 totalSupply

08, Dec 2020
7.13 ms

Line 380-383 in File ERC20PresetMinterPauser.sol

/*@CTK "ERC20 totalSupply"
@tag assume_completion
@post __return == __post._totalSupply */

function totalSupply() public view /*@IGNORE override @IGNORE*/ returns (uint256) {
    return _totalSupply;
}

The code meets the specification.
Formal Verification Request 10

ERC20 balanceOf

08, Dec 2020
8.59 ms

Line 391-394 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "ERC20 balanceOf"
@tag assume_completion
@post __return == __post._balances[account]
*/
```

The code meets the specification.

Formal Verification Request 11

ERC20 allowance

08, Dec 2020
10.36 ms

Line 415-418 in File ERC20PresetMinterPauser.sol

```solidity
function balanceOf(address account) public view /*@IGNORE override
@IGNORE*/ returns (uint256) {
    return _balances[account];
}
```

The code meets the specification.

Formal Verification Request 12

ERC20 transferFrom

08, Dec 2020
892.01 ms

Line 419-421 in File ERC20PresetMinterPauser.sol

```solidity
function allowance(address owner, address spender) public view /*@IGNORE
virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ returns (uint256) {
    return _allowances[owner][spender];
}
```

The code meets the specification.
Line 448-457 in File ERC20PresetMinterPauser.sol

```yaml
/*@CTK "ERC20 transferFrom"
  @tag assume_completion
  @pre sender != address(0)
  @pre recipient != address(0)
  @pre msg.sender != address(0)
  @post sender != recipient -> __post._balances[recipient] ==
  _balances[recipient] + amount
  @post sender != recipient -> __post._balances[sender] ==
  _balances[sender] - amount
  @post sender == recipient -> __post._balances[sender] ==
  _balances[sender]
  @post __post._allowances[sender][msg.sender] ==
  _allowances[sender][msg.sender] - amount
*/
```

Line 458-462 in File ERC20PresetMinterPauser.sol

```yaml
function transferFrom(address sender, address recipient, uint256 amount)
  public /*@IGNORE virtual @IGNORE*/ /*@IGNORE override @IGNORE*/
  returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(),
    _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount
    exceeds allowance");
    return true;
  }
```

✔️ The code meets the specification.

**Formal Verification Request 13**

**ERC20 increaseAllowance**

- 📅 08, Dec 2020
- 🕒 105.61 ms

Line 476-479 in File ERC20PresetMinterPauser.sol

```yaml
/*@CTK "ERC20 increaseAllowance"
  @tag assume_completion
  @post __post._allowances[msg.sender][spender] ==
  _allowances[msg.sender][spender] + addedValue
*/
```

Line 480-483 in File ERC20PresetMinterPauser.sol
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

function increaseAllowance(address spender, uint256 addedValue) public
    /*!IGNORE virtual @IGNORE*/ returns (bool) {
    _approve(_msgSender(), spender, 
        _allowances[_msgSender()][spender].add(addedValue)); 
    return true;
}

The code meets the specification.

Formal Verification Request 14
ERC20 decreaseAllowance

08, Dec 2020
128.16 ms

Line 499-502 in File ERC20PresetMinterPauser.sol

/*@CTK "ERC20 decreaseAllowance"
@tag assume_completion 
@post __post._allowances[msg.sender][spender] == 
_allowances[msg.sender][spender] - subtractedValue */

Line 503-506 in File ERC20PresetMinterPauser.sol

function decreaseAllowance(address spender, uint256 subtractedValue)
    public /*!IGNORE virtual @IGNORE*/ returns (bool) {
    _approve(_msgSender(), spender, 
        _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero");
    return true;
}

The code meets the specification.

Formal Verification Request 15
ERC20 _transfer

08, Dec 2020
265.01 ms

Line 522-529 in File ERC20PresetMinterPauser.sol

/*@CTK "ERC20 _transfer"
@tag assume_completion 
@pre sender != address(0)
@pre recipient != address(0)
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

```
@post sender != recipient -> _balances[recipient] == _balances[recipient] + amount
@post sender != recipient -> _balances[sender] == _balances[sender] - amount
@post sender == recipient -> _balances[sender] == _balances[sender]
```

Line 530-539 in File ERC20PresetMinterPauser.sol

```
function _transfer(address sender, address recipient, uint256 amount) internal virtual {
    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");
    _beforeTokenTransfer(sender, recipient, amount);
    _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
    _balances[recipient] = _balances[recipient].add(amount);
    emit Transfer(sender, recipient, amount);
}
```

The code meets the specification.

**Formal Verification Request 16**

**ERC20 _mint**

📅 08, Dec 2020
⏰ 293.52 ms

Line 550-555 in File ERC20PresetMinterPauser.sol

```
/*@CTK "ERC20 _mint"
@tag assume_completion
@pre account != address(0)
@post ___post.__totalSupply == ___totalSupply + amount
@post ___post.__balances[account] == ___balances[account] + amount

*/
```

Line 556-564 in File ERC20PresetMinterPauser.sol

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");
```
The code meets the specification.

**Formal Verification Request 17**

**ERC20 _burn**

08, Dec 2020

376.15 ms

Line 577-581 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "ERC20 _burn"
@tag assume_completion
@post __post._totalSupply == _totalSupply - amount
@post __post._balances[account] == _balances[account] - amount
*/
```

Line 582-590 in File ERC20PresetMinterPauser.sol

```solidity
function _burn(address account, uint256 amount) internal { /*@IGNORE*/
  require(account != address(0), "ERC20: burn from the zero address");
  _beforeTokenTransfer(account, address(0), amount);
  _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
  _totalSupply = _totalSupply.sub(amount);
  emit Transfer(account, address(0), amount);
}
```

The code meets the specification.

**Formal Verification Request 18**

**ERC20 _approve**

08, Dec 2020

4.55 ms

Line 605-610 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "ERC20 _approve"
@tag assume_completion
@post __post._balances[account] = _balances[account] + amount
*/
```

Line 615-622 in File ERC20PresetMinterPauser.sol

```solidity
function _approve(address account, uint256 value) internal { /*@IGNORE*/
  require(account != address(0), "ERC20: approve from the zero address");
  if (value > _allowances[account].allowance(this, account)) { throw; }
  _allowances[account].allowance(this, account) = value;
  _afterTokenTransfer(account, account, value);
}
```

The code meets the specification.
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

/*@CTK "ERC20 _approve"
@tag assume_completion
@pre owner != address(0)
@pre spender != address(0)
@post __post._allowances[owner][spender] == amount */

Line 611-617 in File ERC20PresetMinterPauser.sol

```solidity
function _approve(address owner, address spender, uint256 amount) internal {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

The code meets the specification.

Formal Verification Request 19
ERC20 _setupDecimals

08, Dec 2020
9.72 ms

Line 626-629 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "ERC20 _setupDecimals"
@tag assume_completion
@post __post._decimals == decimals_
*/

function _setupDecimals(uint8 decimals_) internal {
    _decimals = decimals_
}
```

The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

08, Dec 2020
442.98 ms
Line 679 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_OVERFLOW
```

Line 690-695 in File ERC20PresetMinterPauser.sol

```solidity
function burnFrom(address account, uint256 amount) public /*@IGNORE virtual @IGNORE*/ {
    uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
    _approve(account, _msgSender(), decreasedAllowance);
    _burn(account, amount);
}
```

✅ The code meets the specification.

**Formal Verification Request 21**

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

📅 08, Dec 2020
⏰ 34.79 ms

Line 680 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_BUF_OVERFLOW
```

Line 690-695 in File ERC20PresetMinterPauser.sol

```solidity
function burnFrom(address account, uint256 amount) public /*@IGNORE virtual @IGNORE*/ {
    uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
    _approve(account, _msgSender(), decreasedAllowance);
    _burn(account, amount);
}
```

✅ The code meets the specification.

**Formal Verification Request 22**

Method will not encounter an assertion failure.

📅 08, Dec 2020
⏰ 37.3 ms

Line 681 in File ERC20PresetMinterPauser.sol
The code meets the specification.

**Formal Verification Request 23**

**ERC20Burnable burnFrom**

08, Dec 2020

335.84 ms

The code meets the specification.
**Formal Verification Request 24**

If method completes, integer overflow would not happen.

---

**08, Dec 2020**

6.89 ms

Line 723 in File ERC20PresetMinterPauser.sol

```
723  //@CTK NO_OVERFLOW
```

Line 729-731 in File ERC20PresetMinterPauser.sol

```
729  constructor () internal {
730      _paused = false;
731  }
```

✓ The code meets the specification.

**Formal Verification Request 25**

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

---

**08, Dec 2020**

1.0 ms

Line 724 in File ERC20PresetMinterPauser.sol

```
724  //@CTK NO_BUF_OVERFLOW
```

Line 729-731 in File ERC20PresetMinterPauser.sol

```
729  constructor () internal {
730      _paused = false;
731  }
```

✓ The code meets the specification.

**Formal Verification Request 26**

Method will not encounter an assertion failure.

---

**08, Dec 2020**

1.38 ms

Line 725 in File ERC20PresetMinterPauser.sol

```
725  //@CTK NOASF
```

Line 729-731 in File ERC20PresetMinterPauser.sol
The code meets the specification.

Formal Verification Request 27
Pausable constructor

08, Dec 2020
1.84 ms

The code meets the specification.

Formal Verification Request 28
If method completes, integer overflow would not happen.

08, Dec 2020
8.08 ms

The code meets the specification.
**Formal Verification Request 29**

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

- 08, Dec 2020
- 1.45 ms

Line 737 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_BUF_OVERFLOW
```

Line 742-744 in File ERC20PresetMinterPauser.sol

```solidity
function paused() public view returns (bool) {
    return _paused;
}
```

✔️ The code meets the specification.

**Formal Verification Request 30**

Method will not encounter an assertion failure.

- 08, Dec 2020
- 1.01 ms

Line 738 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO ASF
```

Line 742-744 in File ERC20PresetMinterPauser.sol

```solidity
function paused() public view returns (bool) {
    return _paused;
}
```

✔️ The code meets the specification.

**Formal Verification Request 31**

Pausable paused

- 08, Dec 2020
- 1.03 ms

Line 739-741 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "Pausable paused"
    @post __return == _paused
*/
```
Line 742-744 in File ERC20PresetMinterPauser.sol

```solidity
function paused() public view returns (bool) {
    return _paused;
}
```

✔ The code meets the specification.

**Formal Verification Request 32**
If method completes, integer overflow would not happen.

📅 08, Dec 2020
🕒 22.06 ms

Line 777 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_OVERFLOW
```

Line 784-787 in File ERC20PresetMinterPauser.sol

```solidity
function _pause() internal /*@IGNORE virtual @IGNORE*/ whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}
```

✔ The code meets the specification.

**Formal Verification Request 33**
Buffer overflow / array index out of bound would never happen.

📅 08, Dec 2020
🕒 1.22 ms

Line 778 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_BUF_OVERFLOW
```

Line 784-787 in File ERC20PresetMinterPauser.sol

```solidity
function _pause() internal /*@IGNORE virtual @IGNORE*/ whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}
```

✔ The code meets the specification.
Formal Verification Request 34

Method will not encounter an assertion failure.

08, Dec 2020

1.35 ms

Line 779 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NOASF
```

Line 784-787 in File ERC20PresetMinterPauser.sol

```solidity
function _pause() internal /*@IGNORE virtual @IGNORE*/ whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}
```

The code meets the specification.

Formal Verification Request 35

Pausable _pause

08, Dec 2020

3.72 ms

Line 780-783 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK \"Pausable _pause\"
 tag assume_completion
 @post __post._paused == true
 */
```

Line 784-787 in File ERC20PresetMinterPauser.sol

```solidity
function _pause() internal /*@IGNORE virtual @IGNORE*/ whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}
```

The code meets the specification.

Formal Verification Request 36

If method completes, integer overflow would not happen.

08, Dec 2020

23.02 ms

Line 796 in File ERC20PresetMinterPauser.sol
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

//@CTK NO_OVERFLOW

Line 803-806 in File ERC20PresetMinterPauser.sol

```solidity
function _unpause() internal /*@IGNORE virtual @IGNORE*/ whenPaused {
    _paused = false;
    emit Unpaused(_msgSender());
}
```

The code meets the specification.

**Formal Verification Request 37**

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

📅 08, Dec 2020
🕒 0.99 ms

Line 797 in File ERC20PresetMinterPauser.sol

//@CTK NO_BUF_OVERFLOW

Line 803-806 in File ERC20PresetMinterPauser.sol

```solidity
function _unpause() internal /*@IGNORE virtual @IGNORE*/ whenPaused {
    _paused = false;
    emit Unpaused(_msgSender());
}
```

The code meets the specification.

**Formal Verification Request 38**

Method will not encounter an assertion failure.

📅 08, Dec 2020
🕒 1.06 ms

Line 798 in File ERC20PresetMinterPauser.sol

//@CTK NOASF

Line 803-806 in File ERC20PresetMinterPauser.sol

```solidity
function _unpause() internal /*@IGNORE virtual @IGNORE*/ whenPaused {
    _paused = false;
    emit Unpaused(_msgSender());
}
```

The code meets the specification.
Formal Verification Request 39

Pausable `_unpause`

Line 799-802 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK "Pausable _unpause"
@tag assume_completion
@post __post._paused == false
*/
```

The code meets the specification.

Formal Verification Request 40

If method completes, integer overflow would not happen.

Line 824 in File ERC20PresetMinterPauser.sol

```solidity
/*@CTK NO_OVERFLOW
```

The code meets the specification.

Formal Verification Request 41

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

Line 827-831 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_OVERFLOW
```

The code meets the specification.
Formal Verification Platform for Smart Contracts and Blockchain Ecosystems

Line 825 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_BUF_OVERFLOW

Line 827-831 in File ERC20PresetMinterPauser.sol

```function _beforeTokenTransfer(address from, address to, uint256 amount) internal /*@IGNORE virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ {
super._beforeTokenTransfer(from, to, amount);

```require(!paused(), "ERC20Pausable: token transfer while paused");

```}

The code meets the specification.

Formal Verification Request 42
Method will not encounter an assertion failure.

08, Dec 2020
1.31 ms

Line 826 in File ERC20PresetMinterPauser.sol

```solidity
//@CTK NO_ASMF

Line 827-831 in File ERC20PresetMinterPauser.sol

```function _beforeTokenTransfer(address from, address to, uint256 amount) internal /*@IGNORE virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ {
super._beforeTokenTransfer(from, to, amount);

```require(!paused(), "ERC20Pausable: token transfer while paused");

```}

The code meets the specification.

Formal Verification Request 43
Context _msgSender

08, Dec 2020
7.34 ms

Line 16-19 in File ERC20PresetMinterPauser_Context.sol

```solidity
/*@CTK "Context _msgSender"
@tag assume_completion
@post __return == msg.sender
*/
Line 20-22 in File ERC20PresetMinterPauser_Context.sol

```solidity
function _msgSender() internal view /*@IGNORE virtual @IGNORE*/ returns (address payable) {
    return msg.sender;
}
```

✔️ The code meets the specification.

**Formal Verification Request 44**

**Context _msgData**

![Date Icon] 08, Dec 2020

8.14 ms

Line 23-26 in File ERC20PresetMinterPauser_Context.sol

```solidity
/*@CTK "Context _msgData"
@tag assume_completion
@post __return == msg.data
*/
```

Line 27-30 in File ERC20PresetMinterPauser_Context.sol

```solidity
function _msgData() internal view /*@IGNORE virtual @IGNORE*/ returns (bytes memory) {
    this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
}
```

✔️ The code meets the specification.
Source Code with CertiK Labels

ERC20PresetMinterPauser.sol

```solidity
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

import './ERC20PresetMinterPauser_EnumerableSet.sol';
import './ERC20PresetMinterPauser_Address.sol';
import './ERC20PresetMinterPauser_Context.sol';
import './ERC20PresetMinterPauser_AccessControl.sol';

/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an
 * entire class of bugs, so it's recommended to use it always.
 */
library SafeMath {

/**
 * @dev Returns the addition of two unsigned integers, reverting on
 * overflow.
 *
 * Counterpart to Solidity's `+` operator.
 *
 * Requirements:
 *
 * - Addition cannot overflow.
 */
/*@CTK "SafeMath add"
	@tag assume_completion
	@post (a + b < a || a + b < b) == __reverted
	@post !__reverted -> __return == a + b
	@post !__reverted -> !__has_overflow
	@post !__reverted -> !__has_assertion_failure
	@post !(__has_buf_overflow)
```
```solidity
41 */
42 function add(uint256 a, uint256 b) internal pure returns (uint256) {
43     uint256 c = a + b;
44     require(c >= a, "SafeMath: addition overflow");
45
46     return c;
47 }
48
49 /**
50 * @dev Returns the subtraction of two unsigned integers, reverting on
51 * overflow (when the result is negative).
52 * Counterpart to Solidity's `-` operator.
53 * Requirements:
54 * - Subtraction cannot overflow.
55 */
56 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
57     return sub(a, b, "SafeMath: subtraction overflow");
58 }
59
60 /**
61 * @dev Returns the subtraction of two unsigned integers, reverting
62 * with custom message on
63 * overflow (when the result is negative).
64 * Counterpart to Solidity's `-` operator.
65 * Requirements:
66 * - Subtraction cannot overflow.
67 */
68 @CTK "SafeMath sub"
69 @tag assume_completion
70 @post (a < b) == __reverted
71 @post !__reverted -> __return == a - b
72 @post !__reverted -> !__has_overflow
73 @post !__reverted -> !__has_assertion_failure
74 @post !(__has_buf_overflow)
75 */
76 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
77     require(b <= a, errorMessage);
78     uint256 c = a - b;
79
80     return c;
81 ```
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
    if (a == 0) {
        return 0;
    }

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;
}
* Requirements:

* - The divisor cannot be zero.

/**<*
@dev Returns the integer division of two unsigned integers. Reverts with custom message on
* division by zero. The result is rounded towards zero.
* Counterpart to Solidity's `÷` operator. Note: this function uses a
* `revert` opcode (which leaves remaining gas untouched) while
Solidity
* uses an invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.

/**<*/
/*@CTK "SafeMath div"
@tag assume_completion
@post (b <= 0) == __reverted
@post !__reverted -> __return == a / b
@post !__reverted -> !_has_overflow
@post !__reverted -> !_has_assertion_failure
@post !(__has_buf_overflow)
/**<*/
function div(uint256 a, uint256 b, string memory errorMessage) internal
pure returns (uint256) {
    require(b > 0, errorMessage);
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this
doesn't hold

    return c;
}

/**
@dev Returns the remainder of dividing two unsigned integers.
(unsigned integer modulo),
* Reverts when dividing by zero.
*
* Counterpart to Solidity's `%` operator. This function uses a
`revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an invalid opcode to revert (consuming all remaining gas).
* Requirements:
  - The divisor cannot be zero.
*/

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

/**
 * @dev Returns the remainder of dividing two unsigned integers.
 * Reverts with custom message when dividing by zero.
 * Counterpart to Solidity's `\%` operator. This function uses a `revert`
 * opcode (which leaves remaining gas untouched) while Solidity uses an invalid opcode to revert (consuming all remaining gas).
 * Requirements:
  - The divisor cannot be zero.
*/

```solidity
/*@CTK "SafeMath mod"
@tag assume_completion
@post b != 0 -> !__reverted
@post !__reverted -> __return == a % b
@post !__reverted -> !__has_overflow
@post !(__has_buf_overflow)
@post !(__has_assertion_failure)
*/

function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
    require(b != 0, errorMessage);
    return a % b;
}
```

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
*/

```solidity
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
```
function totalSupply() external view returns (uint256);

/**
 * @dev Returns the amount of tokens owned by `account`.
 */
function balanceOf(address account) external view returns (uint256);

/**
 * @dev Moves `amount` tokens from the caller's account to `recipient`.
 * Returns a boolean value indicating whether the operation succeeded.
 * Emits a {Transfer} event.
 */
function transfer(address recipient, uint256 amount) external returns (bool);

/**
 * @dev Returns the remaining number of tokens that `spender` will be
 * allowed to spend on behalf of `owner` through {transferFrom}. This
 * is zero by default.
 * This value changes when {approve} or {transferFrom} are called.
 */
function allowance(address owner, address spender) external view returns (uint256);

/**
 * @dev Sets `amount` as the allowance of `spender` over the caller's
 * tokens.
 * Returns a boolean value indicating whether the operation succeeded.
 * IMPORTANT: Beware that changing an allowance with this method brings
 * the risk that someone may use both the old and the new allowance by
 * unfortunate transaction ordering. One possible solution to mitigate this race
 * condition is to first reduce the spender's allowance to 0 and set the
 * desired value afterwards:
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 * Emits an {Approval} event.
 */
function approve(address spender, uint256 amount) external returns (bool);
```solidity
/**
* \@dev Moves `amount` tokens from `sender` to `recipient` using the
* allowance mechanism. `amount` is then deducted from the caller's
* allowance.
* Returns a boolean value indicating whether the operation succeeded.
* \@dev Emits a \{Transfer\} event.
*/
function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

/**
* \@dev Emitted when `value` tokens are moved from one account (`from`) to
* another (`to`).
* Note that `value` may be zero.
*/
event Transfer(address indexed from, address indexed to, uint256 value);

/**
* \@dev Emitted when the allowance of a `spender` for an `owner` is set
* by a call to \{approve\}. `value` is the new allowance.
*/
event Approval(address indexed owner, address indexed spender, uint256 value);

/**
* \@dev Implementation of the \{IERC20\} interface.
* This implementation is agnostic to the way tokens are created. This
* means that a supply mechanism has to be added in a derived contract using
* \{_mint\}.
* For a generic mechanism see \{ERC20PresetMinterPauser\}.
* \@dev TIP: For a detailed writeup see our guide
* \(\text{https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms/226}\) to implement supply mechanisms.
* \@dev We have followed general OpenZeppelin guidelines: functions revert
* instead
*/
* of returning `false` on failure. This behavior is nonetheless conventional and does not conflict with the expectations of ERC20 applications.

* Additionally, an `{Approval}` event is emitted on calls to `{transferFrom}`. This allows applications to reconstruct the allowance for all accounts just by listening to said events. Other implementations of the EIP may not emit these events, as it isn't required by the specification.

* Finally, the non-standard `{decreaseAllowance}` and `{increaseAllowance}` functions have been added to mitigate the well-known issues around setting allowances. See `{IERC20-approve}`.

```solidity
contract ERC20 is Context, IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    /**
    * @dev Sets the values for {name} and {symbol}, initializes {decimals} with a default value of 18.
    * To select a different value for {decimals}, use {_setupDecimals}.
    * All three of these values are immutable: they can only be set once during construction.
    */
    constructor (string memory name_, string memory symbol_) public {
        _name = name_;
        _symbol = symbol_;
        _decimals = 18;
    }
```
/**
 * @dev Returns the name of the token.
 */
/*@CTK "ERC20 name"
@tag assume_completion
@post __return == __post._name
*/
function name() public view returns (string memory) {
    return _name;
}

/**
 * @dev Returns the symbol of the token, usually a shorter version of the
 * name.
 */
/*@CTK "ERC20 symbol"
@tag assume_completion
@post __return == __post._symbol
*/
function symbol() public view returns (string memory) {
    return _symbol;
}

/**
 * @dev Returns the number of decimals used to get its user
 * representation.
 * For example, if `decimals` equals `2`, a balance of `505` tokens
 * should be displayed to a user as `5,05` (`505 / 10 ** 2`).
 * Tokens usually opt for a value of 18, imitating the relationship
 * between Ether and Wei. This is the value {ERC20} uses, unless
 * `_setupDecimals` is
 * called.
 * NOTE: This information is only used for _display_ purposes: it in
 * no way affects any of the arithmetic of the contract, including
 * {IERC20-balanceOf} and {IERC20-transfer}.
 */
/*@CTK "ERC20 decimals"
@tag assume_completion
@post __return == __post._decimals
*/
function decimals() public view returns (uint8) {
    return _decimals;
}
* Requirements:

* - `spender` cannot be the zero address.

*/

_approve(_msgSender(), spender, amount);

return true;

}
* This is an alternative to {approve} that can be used as a mitigation for
  * problems described in {IERC20-approve}.
  * Emits an {Approval} event indicating the updated allowance.
  * Requirements:
  * - `spender` cannot be the zero address.
*/
/*@CTK "ERC20 increaseAllowance"
@tag assume_completion
@post __post._allowances[msg.sender][spender] ==
_allowances[msg.sender][spender] + addedValue */
_approve(_msgSender(), spender,
_allowances[_msgSender()][spender].add(addedValue));
return true;
}

/**
 * @dev Atomically decreases the allowance granted to `spender` by the caller.
 * This is an alternative to {approve} that can be used as a mitigation for
 * problems described in {IERC20-approve}.
 * Emits an {Approval} event indicating the updated allowance.
 * Requirements:
 * - `spender` cannot be the zero address.
 * - `spender` must have allowance for the caller of at least
 *   `subtractedValue`.
 */
/*@CTK "ERC20 decreaseAllowance"
@tag assume_completion
@post __post._allowances[msg.sender][spender] ==
_allowances[msg.sender][spender] - subtractedValue */
_approve(_msgSender(), spender,
_allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero");
return true;
/**
 * @dev Moves tokens `amount` from `sender` to `recipient`.
 * This is internal function is equivalent to {transfer}, and can be used to
 * e.g. implement automatic token fees, slashing mechanisms, etc.
 * Emits a {Transfer} event.
 *
 * Requirements:
 * - `sender` cannot be the zero address.
 * - `recipient` cannot be the zero address.
 * - `sender` must have a balance of at least `amount`.
 */

/*@CTK "ERC20 _transfer"
@tag assume_completion
@pre sender != address(0)
@pre recipient != address(0)
@post sender != recipient -> __post._balances[recipient] ==
→ _balances[recipient] + amount
@post sender != recipient -> __post._balances[sender] ==
→ _balances[sender] - amount
@post sender == recipient -> __post._balances[sender] ==
→ _balances[sender]
*/

require(sender != address(0), "ERC20: transfer from the zero address");
require(recipient != address(0), "ERC20: transfer to the zero address");

_beforeTokenTransfer(sender, recipient, amount);

_balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
_balances[recipient] = _balances[recipient].add(amount);
emit Transfer(sender, recipient, amount);

/** @dev Creates `amount` tokens and assigns them to `account`, increasing*
 * the total supply.
 * Emits a {Transfer} event with `from` set to the zero address.
 * Requirements:
 * */
/* - `to` cannot be the zero address. */

/*@CTK "ERC20 _mint"
@tag assume_completion
@pre account != address(0)
@post __post._totalSupply == _totalSupply + amount
@post __post._balances[account] == _balances[account] + amount
*/

require(account != address(0), "ERC20: mint to the zero address");

_beforeTokenTransfer(address(0), account, amount);

_totalSupply = _totalSupply.add(amount);
_balances[account] = _balances[account].add(amount);
emit Transfer(address(0), account, amount);

/**
 * @dev Destroys `amount` tokens from `account`, reducing the
 * total supply.
 * * Emits a {Transfer} event with `to` set to the zero address.
 * * Requirements:
 * - `account` cannot be the zero address.
 * - `account` must have at least `amount` tokens.
 */

/*@CTK "ERC20 _burn"
@tag assume_completion
@post __post._totalSupply == _totalSupply - amount
@post __post._balances[account] == _balances[account] - amount
*/

require(account != address(0), "ERC20: burn from the zero address");

_beforeTokenTransfer(account, address(0), amount);

_balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
_totalSupply = _totalSupply.sub(amount);
emit Transfer(account, address(0), amount);

/**
 * @dev Sets `amount` as the allowance of `spender` over the `owner`s
 * tokens.
 */
* This internal function is equivalent to `approve`, and can be used to
  * e.g. set automatic allowances for certain subsystems, etc.
  * Emits an {Approval} event.
  * Requirements:
  * - `owner` cannot be the zero address.
  * - `spender` cannot be the zero address.
  */
/*@CTK "ERC20 _approve"
@tag assume_completion
@pre owner != address(0)
@pre spender != address(0)
@post __post._allowances[owner][spender] == amount
*/
require(owner != address(0), "ERC20: approve from the zero
dress");
require(spender != address(0), "ERC20: approve to the zero
dress");

_allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}

/**
 * @dev Sets {decimals} to a value other than the default one of 18.
 * WARNING: This function should only be called from the constructor.
 * Most applications that interact with token contracts will not expect
 * {decimals} to ever change, and may work incorrectly if it does.
 */
/*@CTK "ERC20 _setupDecimals"
@tag assume_completion
@post __post._decimals == decimals_
*/
function _setupDecimals(uint8 decimals_) internal {
  _decimals = decimals_;
}

 /**
 * @dev Hook that is called before any transfer of tokens. This
 * includes
 * minting and burning.
 *
* Calling conditions:

  * - when `from` and `to` are both non-zero, `amount` of `from`'s tokens will be transferred to `to`.
  * - when `from` is zero, `amount` tokens will be minted for `to`.
  * - when `to` is zero, `amount` of `from`'s tokens will be burned.
  * - `from` and `to` are never both zero.

* To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].

```solidity
/**
 * @dev Extension of {ERC20} that allows token holders to destroy both their own
 * tokens and those that they have an allowance for, in a way that can be recognized off-chain (via event analysis).
 */
using SafeMath for uint256;

/**
 * @dev Destroys `amount` tokens from the caller.
 * See {ERC20-_burn}.
 */
_burn(_msgSender(), amount);

/**
 * @dev Destroys `amount` tokens from `account`, deducting from the caller's allowance.
 * See {ERC20-_burn} and {ERC20-allowance}.
 * Requirements:
 * - the caller must have allowance for `accounts`'s tokens of at least `amount`.
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "ERC20Burnable burnFrom"
```
@tag assume_completion
@post account != address(0)
  @post amount <= _balances[account] && amount <=
  _allowances[account][msg.sender]
  @post __post._balances[account] == _balances[account] - amount
  @post __post._totalSupply == _totalSupply - amount
  @post __post._allowances[account][msg.sender] ==
  _allowances[account][msg.sender] - amount
  /*
  uint256 decreasedAllowance = allowance(account,
  _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
  _approve(account, _msgSender(), decreasedAllowance);
  _burn(account, amount);
  */
/**
 * @dev Contract module which allows children to implement an emergency
 * stop mechanism that can be triggered by an authorized account.
 * This module is used through inheritance. It will make available the
 * modifiers `whenNotPaused` and `whenPaused`, which can be applied to
 * the functions of your contract. Note that they will not be pausable by
 * simply including this module, only once the modifiers are put in place.
 */
/**
 * @dev Emitted when the pause is triggered by `account`.
 */
event Paused(address account);
/**
 * @dev Emitted when the pause is lifted by `account`.
 */
event Unpaused(address account);

bool private _paused;
/**
 * @dev Initializes the contract in unpaused state.
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "Pausable constructor"
  @post __post._paused == false
*/
constructor () internal {
    _paused = false;
}

/**
 * @dev Returns true if the contract is paused, and false otherwise.
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "Pausable paused"
@post __return == _paused
*/
function paused() public view returns (bool) {
    return _paused;
}

/**
 * @dev Modifier to make a function callable only when the contract is
 * not paused.
 * Requirements:
 * - The contract must not be paused.
 */
modifier whenNotPaused() {
    require(!_paused, "Pausable: paused");
    _;
}

/**
 * @dev Modifier to make a function callable only when the contract is
 * paused.
 * Requirements:
 * - The contract must be paused.
 */
modifier whenPaused() {
    require(_paused, "Pausable: not paused");
    _;
}

/**
 * @dev Triggers stopped state.
 */

page 48
* Requirements:

* - The contract must not be paused.

/*@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "Pausable _pause"
@tag assume_completion
@post __post._paused == true
*/
Paused(_msgSender());

/**
* @dev Returns to normal state.
*
* Requirements:
*
* - The contract must be paused.

/*@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "Pausable _unpause"
@tag assume_completion
@post __post._paused == false
*/
Unpaused(_msgSender());

/**
* @dev ERC20 token with pausable token transfers, minting and burning.
*
* Useful for scenarios such as preventing trades until the end of an evaluation
* period, or having an emergency switch for freezing all token transfers in the
* event of a large bug.

/*@dev See {ERC20-_beforeTokenTransfer}.
*/
* - the contract must not be paused.

```solidity
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
super._beforeTokenTransfer(from, to, amount);

require(!paused(), "ERC20Pausable: token transfer while paused");
```

```solidity
/**
 * @dev {ERC20} token, including:
 * - ability for holders to burn (destroy) their tokens
 * - a minter role that allows for token minting (creation)
 * - a pauser role that allows to stop all token transfers
 * This contract uses {AccessControl} to lock permissioned functions using
 * different roles - head to its documentation for details.
 * The account that deploys the contract will be granted the minter and
 * pauser roles, as well as the default admin role, which will let it grant both
 * minter and pauser roles to other accounts.
 */
contract ERC20PresetMinterPauser is Context, AccessControl, ERC20Burnable, ERC20Pausable {
    bytes32 public constant MINTER_ROLE = 0; // keccak256("MINTER_ROLE");
    bytes32 public constant PAUSER_ROLE = 0; // keccak256("PAUSER_ROLE");

    /**
     * @dev Grants `DEFAULT_ADMIN_ROLE`, `MINTER_ROLE` and `PAUSER_ROLE` to
     * the account that deploys the contract.
     * See {ERC20-constructor}.
     */
    constructor(string memory name, string memory symbol) public ERC20(name, symbol) {
        _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
        _setupRole(MINTER_ROLE, _msgSender());
        _setupRole(PAUSER_ROLE, _msgSender());
    }
```
** @dev Creates `amount` new tokens for `to`.

* See {ERC20-_mint}.

* Requirements:

* - the caller must have the `MINTER_ROLE`.

*/
require(hasRole(MINTER_ROLE, _msgSender()), "ERC20PresetMinterPauser: must have minter role to mint");
_mint(to, amount);

/**
 * @dev Pauses all token transfers.
 * See {ERC20Pausable} and {Pausable-_pause}.

* Requirements:

* - the caller must have the `PAUSER_ROLE`.

*/
require(hasRole(PAUSER_ROLE, _msgSender()), "ERC20PresetMinterPauser: must have pauser role to pause");
_pause();

/**
 * @dev Unpauses all token transfers.
 * See {ERC20Pausable} and {Pausable-_unpause}.

* Requirements:

* - the caller must have the `PAUSER_ROLE`.

*/
require(hasRole(PAUSER_ROLE, _msgSender()), "ERC20PresetMinterPauser: must have pauser role to unpause");
_unpause();

super._beforeTokenTransfer(from, to, amount);
ERC20PresetMinterPauser_Context.sol

```solidity
pragma solidity >=0.6.0 <0.8.0;

/*
 * @dev Provides information about the current execution context, including
 * the
 * sender of the transaction and its data. While these are generally
 * available
 * via msg.sender and msg.data, they should not be accessed in such a
 * direct
 * manner, since when dealing with GSN meta-transactions the account
 * sending and
 * paying for execution may not be the actual sender (as far as an
 * application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */

/*@CTK "Context _msgSender"
@tag assume_completion
@post __return == msg.sender
*/
return msg.sender;

/*@CTK "Context _msgData"
@tag assume_completion
@post __return == msg.data
*/
this; // silence state mutability warning without generating
// bytecode - see https://github.com/ethereum/solidity/issues/2691
return msg.data;
```