SMART CONTRACT AUDIT REPORT

For

TYME Token (TYME)

Prepared By: SFI Team

Prepared for: TYME Team

Prepared on: 19/11/2021

Table of Content

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and

does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SaferICO) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

- # Overview of the audit

The project has 1 file. The main file contains approx 757 lines of Solidity code. Most of the functions and state variables are well commented on using the Nat spec documentation, but that does not create any vulnerability.

- # Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices automatically.

1. Unit tests passing.

2. Compilator warnings;

3. Race Conditions. Reentrancy. Cross-function Race Conditions. Pitfalls in Race Condition solutions;

4. Possible delays in data delivery;

5. Transaction-Ordering Dependence (front running);

6. Timestamp Dependence;

7. Integer Overflow and Underflow;

8. DoS with (unexpected) Revert;

9. DoS with Block Gas Limit;

10. Call Depth Attack.

11. Methods execution permissions;

12. Oracles calls;

13. Economy model. It’s important to forecast scenarios when a user is provided with additional economic motivation or faced with limitations. If application logic is based on incorrect economy model, the application will not function correctly and participants will incur financial losses. This type of issue is most often found in bonus rewards systems.
14. The impact of the exchange rate on the logic;
15. Private user data leaks.

- # Good things in smart contract

Compiler version is static: -
=> In this file, you have put “pragma solidity 0.8.0;” which is a good way to define the compiler version.

```
pragma solidity 0.8.0;
```

- **openzeppelin library: -**

TYME is using openzeppelin library it is a good thing. All contract are based on openzeppelin library which develops by professional developers and it is one of the most secured library in the blockchain industry

```
File: @openzeppelin/contracts/utils/Context.sol
File: @openzeppelin/contracts/access/Ownable.sol
File: contracts/token/BEP20/lib/IBEP20.sol
File: contracts/token/BEP20/lib/BEP20.sol
File: contracts/token/BEP20/lib/BEP20Capped.sol
File: contracts/token/BEP20/lib/BEP20Mintable.sol
File: contracts/token/BEP20/lib/BEP20Burnable.sol
File: contracts/token/BEP20/CommonBEP20.sol
```

- **Ownable library : -**
  - Here you TYME token using ownable library, Initializes the contract setting the deployer as the initial owner

```
abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed
previousOwner, address indexed newOwner);

    /**
     * @dev Initializes the contract setting the
deployer as the initial owner.
     */
    constructor () {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0),
msgSender);
    }
```

Here you TYME token using BEP20Mintable library,

Implementation of the BEP20Mintable. Extension of {BEP20} that adds a minting behavior.

```
abstract contract BEP20Mintable is BEP20 {
    // indicates if minting is finished
    bool private _mintingFinished = false;


    event MintFinished();

    modifier canMint() {
        require(!_mintingFinished, "BEP20Mintable: minting
is finished");
        _;
    }
```

- Here you TYME using BEP20Burnable librariy Extension of {BEP20} 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).

```
abstract contract BEP20Burnable is BEP20 {

    function burn(uint256 amount) public virtual {
        _burn(_msgSender(), amount);
      }
function burnFrom(address account, uint256 amount)
  public virtual {
       uint256 currentAllowance =
  allowance(account, _msgSender());
       require(currentAllowance >= amount, "BEP20:
  burn amount exceeds allowance");
       _approve(account, _msgSender(),
  currentAllowance - amount);
       _burn(account, amount);
    }
  }
```

• Critical vulnerabilities found in the contract

There not Critical severity vulnerabilities found

• High vulnerabilities found in the contract

There not High severity vulnerabilities found

• Medium vulnerabilities found in the contract

There not Medium severity vulnerabilities found

• Low severity vulnerabilities found

#Gas costs:

```
  function name() public view override returns (string memory) {
        return _name;
    }
function symbol() public view override returns (string memory) {
        return _symbol;
    }
```

In detail

Gas requirement of function BEP20.name and symbol are infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)

#No return:

```
function name() external view returns (string memory);
    function symbol() external view returns (string memory);
function decimals() external view returns (uint8);
 function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function getOwner() external view returns (address);
    function transfer(address recipient, uint256 amount) external
returns (bool);

    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);

    function approve(address spender, uint256 amount) external returns
(bool);
    function allowance(address _owner, address spender) external view
returns (uint256);
```

In detail

IBEP20: Defines a return type but never explicitly returns a value.

- **Notes**

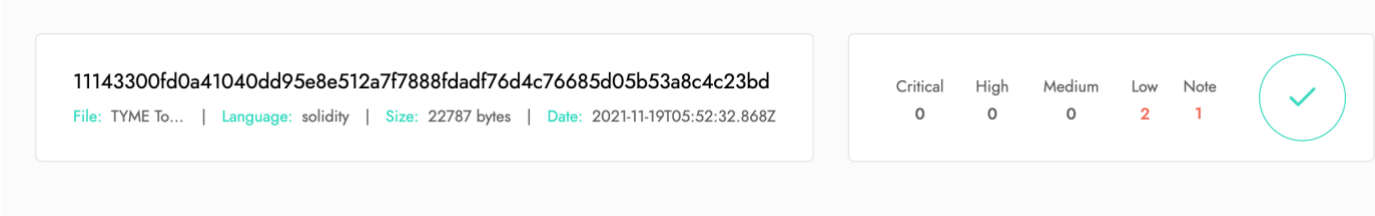
#Similar variable names:

```
   constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
        _decimals = 18;
    }
```

BEP20.(string,string) : Variables have very similar names "_name" , “symbol” and "name_","symbol".
Note: Modifiers are currently not considered by this static analysis.

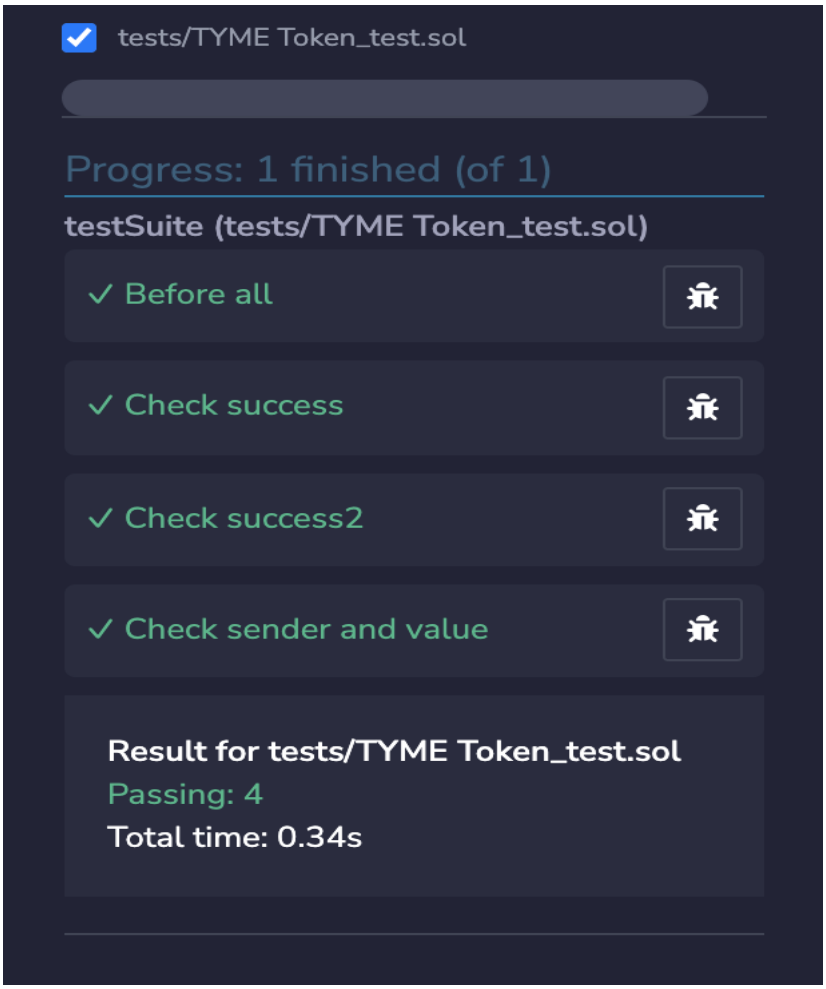
Testing proves:

1- Check for security

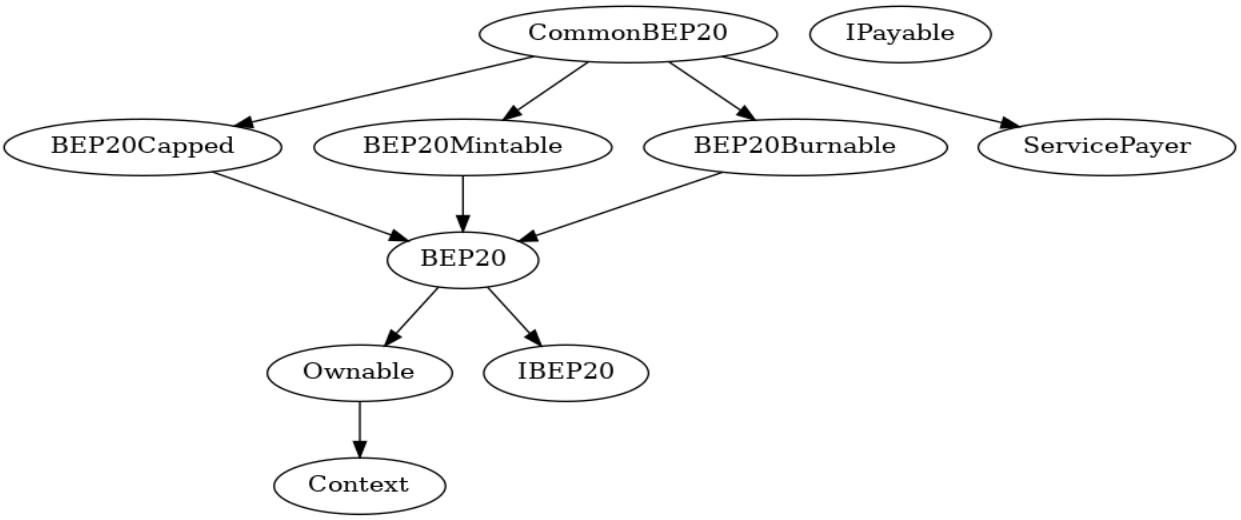


2- SOLIDITY STATIC ANALYSIS

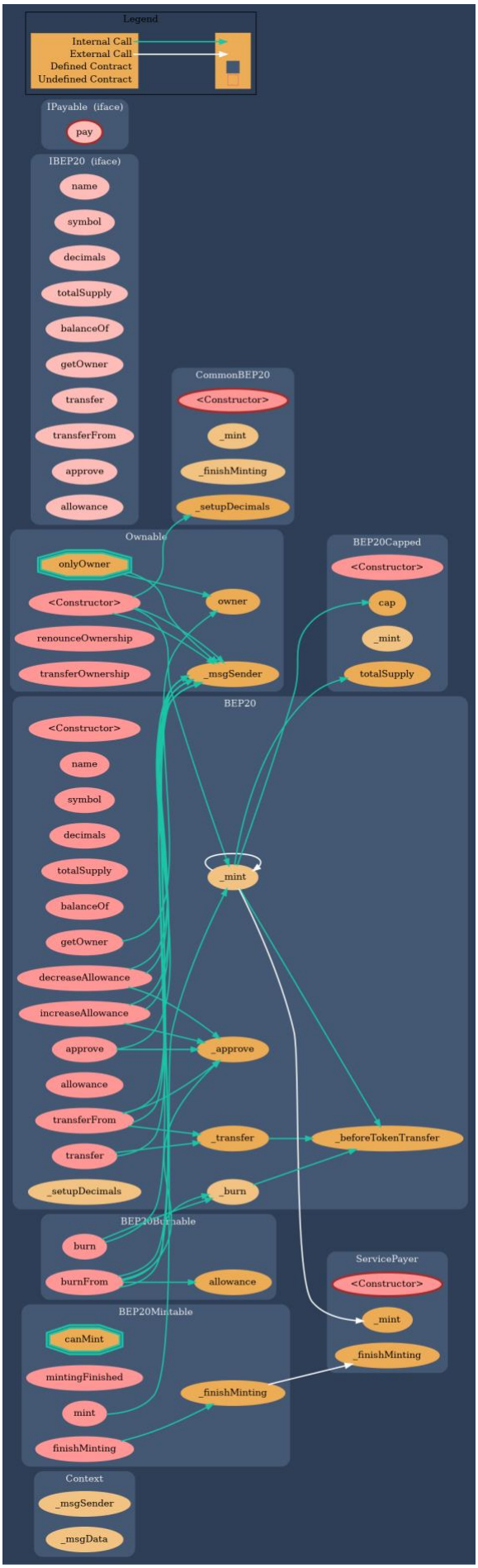
3- SOLIDITY UNIT TESTING



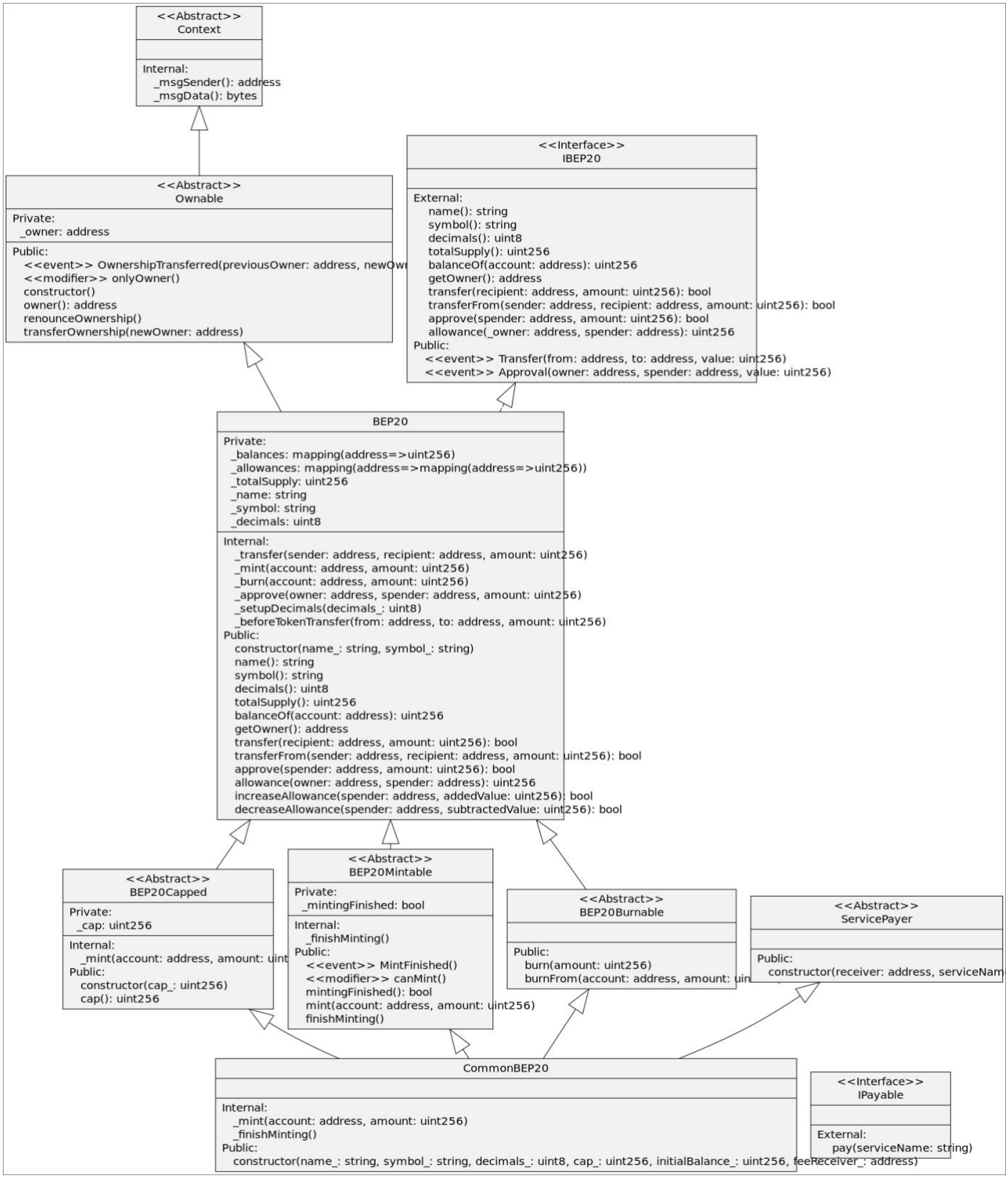
4- Inheritance graph



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IPayable (iface)
pay
IBEP20 (iface)
name
symbol
decimals
totalSupply
balanceOf
getOwner
transfer
transferFrom
approve
allowance
CommonBEP20
<Constructor>
_mint
_finishMinting
_setupDecimals
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
owner
_msgSender
BEP20Capped
<Constructor>
cap
_mint
totalSupply
BEP20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
getOwner
decreaseAllowance
increaseAllowance
approve
allowance
transferFrom
transfer
_setupDecimals
_mint
_approve
_transfer
_beforeTokenTransfer
_burn
BEP20Burnable
burn
burnFrom
allowance
ServicePayer
<Constructor>
_mint
_finishMinting
BEP20Mintable
canMint
mintingFinished
mint
finishMinting
_finishMinting
Context
_msgSender
_msgData

- **Unified Modeling Language (UML)**



Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/TYME Token.sol	721a67afc577402318fdc7aed790c4ed257cf7be

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
IBEP20	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
BEP20	Implementation	Ownable, IBEP20		

- | └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
- | └ | name | Public ❗ | |NO❗ |
- | └ | symbol | Public ❗ | |NO❗ |
- | └ | decimals | Public ❗ | |NO❗ |
- | └ | totalSupply | Public ❗ | |NO❗ |
- | └ | balanceOf | Public ❗ | |NO❗ |
- | └ | getOwner | Public ❗ | |NO❗ |
- | └ | transfer | Public ❗ | 🛑 |NO❗ |
- | └ | transferFrom | Public ❗ | 🛑 |NO❗ |
- | └ | approve | Public ❗ | 🛑 |NO❗ |
- | └ | allowance | Public ❗ | |NO❗ |
- | └ | increaseAllowance | Public ❗ | 🛑 |NO❗ |
- | └ | decreaseAllowance | Public ❗ | 🛑 |NO❗ |
- | └ | _transfer | Internal 🔒 | 🛑 | |
- | └ | _mint | Internal 🔒 | 🛑 | |
- | └ | _burn | Internal 🔒 | 🛑 | |
- | └ | _approve | Internal 🔒 | 🛑 | |
- | └ | _setupDecimals | Internal 🔒 | 🛑 | |
- | └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |
- ||||||
- | **BEP20Capped** | Implementation | BEP20 |||
- | └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
- | └ | cap | Public ❗ | |NO❗ |
- | └ | _mint | Internal 🔒 | 🛑 | |
- ||||||
- | **BEP20Mintable** | Implementation | BEP20 |||
- | └ | mintingFinished | Public ❗ | |NO❗ |
- | └ | mint | Public ❗ | 🛑 | canMint |
- | └ | finishMinting | Public ❗ | 🛑 | canMint |
- | └ | _finishMinting | Internal 🔒 | 🛑 | |
- ||||||
- | **BEP20Burnable** | Implementation | BEP20 |||
- | └ | burn | Public ❗ | 🛑 |NO❗ |
- | └ | burnFrom | Public ❗ | 🛑 |NO❗ |
- ||||||
- | **IPayable** | Interface | |||
- | └ | pay | External ❗ | 💵 |NO❗ |
- ||||||
- | **ServicePayer** | Implementation | |||

```
| └ | <Constructor> | Public ❗ |  💵 |NO❗ |
||||||
| **CommonBEP20** | Implementation | BEP20Capped, BEP20Mintable, BEP20Burnable, ServicePayer |||
| └ | <Constructor> | Public ❗ |  💵 | BEP20 BEP20Capped ServicePayer |
| └ | _mint | Internal 🔒 | 🛑 | onlyOwner |
| └ | _finishMinting | Internal 🔒 | 🛑 | onlyOwner |


 Legend

| Symbol | Meaning |
|:--------:|-----------|
|  🛑  | Function can modify state |
|  💵  | Function is payable |
```

- # Summary of the Audit

According to automatically test, the customer`s solidity smart contract is **Secured**.

The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

The test found 0 critical, 0 high, 0 medium, 2 low issues, and 1 note.