



SMART CONTRACT AUDIT REPORT

For

Shiba MetaWar (SHM)

<https://shibametawar.io/>

Prepared By: SFI Team

Prepared for: SHM team

Prepared on: 22/11/2021

Table of Content

- Disclaimer
- Overview of the audit
- Attacks made to the contract
- Good things in smart contract
- Critical vulnerabilities found in the contract
- High vulnerabilities found in the contract
- Medium vulnerabilities found in the contract
- Low severity vulnerabilities found in the contract
- Notes
- Testing proves
- Automatic general report
- Summary of the audit

- **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. It contains approx 637 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. Compiler 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. Not relevant in modern ethereum network
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 0.8.2;” which is a good way to define the compiler version.

```
pragma solidity 0.8.2;
```

- **openzeppelin library: -**

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

```
@openzeppelin/contracts/utils/Context.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/IERC20.:
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol
```

- **Ownable library : -**

- Here you SHM 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() {
        _setOwner(_msgSender());
    }
    function owner() public view virtual returns
(address) {
        return _owner;
    }
}
```

Here you SHM token using ERC20 library,
 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}`. 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}`.

```
contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256))
private _allowances;
    uint256 private _totalSupply;
    string private _name;
    string private _symbol;
```

- o Here you SHM using ERC20Burnable library 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).

```
abstract contract ERC20Burnable is Context, ERC20 {
    /**
     * @dev Destroys `amount` tokens from the
     * caller.
     *
     * See {ERC20-_burn}.
     */
    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, "ERC20:
        burn amount exceeds allowance");
        unchecked {
            _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**

#Similar variable names:

```
_name = name_;  
_symbol = symbol_;
```

In detail

ERC20.(string,string) : Variables have very similar names "_symbol" and "symbol_". Note: Modifiers are currently not considered by this static analysis.

Constant/View/Pure functions:

```
function _afterTokenTransfer(  
    address from,  
    address to,  
    uint256 amount  
) internal virtual {}  
}
```

In detail

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

- **Notes**

#Guard conditions:

```
require(owner() == _msgSender(), "Ownable: caller is not the owner");
require(newOwner != address(0), "Ownable: new owner is the zero address");
```

In detail

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

#NO return:

```
function totalSupply() external view
returns (uint256);
```

In detail

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.

Testing proves:

1- Check for security

ddbd626af4f732982f2d96f27e11aa8e6ffec079a92edf6d8dc346095e6e73dc
File: SHMTok... | Language: solidity | Size: 19766 bytes | Date: 2021-11-22T09:13:20.062Z

Critical	High	Medium	Low	Note
0	0	0	2	2



2- SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

- ERC
 - Select ERC
 - ERC20:
 - 'decimals' should be 'uint8'
- Miscellaneous
 - Select Miscellaneous
 - Constant/View/Pure functions:
 - Potentially constant/view/pure functions
 - Similar variable names:
 - Variable names are too similar
 - No return:
 - Function with 'returns' not returning
 - Guard conditions:
 - Ensure appropriate use of require/assert
 - Result not used:
 - The result of an operation not used
 - String length:
 - Bytes length != String length
 - Delete from dynamic array:
 - 'delete' leaves a gap in array
 - Data truncated:
 - Division on int/uint values truncates the result

SOLIDITY STATIC ANALYSIS

☒ Select all ☒ Autorun Run





- Security
 - Select Security
 - Transaction origin:
 - 'tx.origin' used
 - Check-effects-interaction:
 - Potential reentrancy bugs
 - Inline assembly:
 - Inline assembly used
 - Block timestamp:
 - Can be influenced by miners
 - Low level calls:
 - Should only be used by experienced devs
 - Block hash:
 - Can be influenced by miners
 - Selfdestruct:
 - Contracts using destructured contract can be broken
- Gas & Economy
 - Select Gas & Economy
 - Gas costs:
 - Too high gas requirement of functions
 - This on local calls:
 - Invocation of local functions via 'this'
 - Delete dynamic array:
 - Use require/assert to ensure complete deletion
 - For loop over dynamic array:
 - Iterations depend on dynamic array's size
 - Ether transfer in loop:
 - Transferring Ether in a for/while/do-while loop

3- SOLIDITY UNIT TESTING

✓ tests/SHMToken_test.sol

Progress: 1 finished (of 1)

PASS testSuite (tests/SHMToken_test.sol)

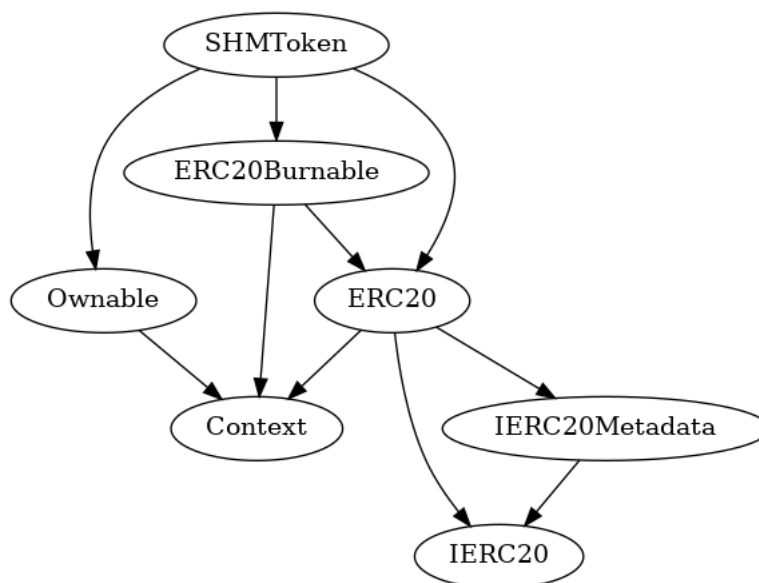
- ✓ Before all 
- ✓ Check success 
- ✓ Check success2 
- ✓ Check sender and value 

Result for tests/SHMToken_test.sol

Passing: 4

Total time: 0.35s

4- Inheritance graph



5- Call graph

• Automatic general report

• Files Description Table

•

• | File Name | SHA-1 Hash |

• |-----|-----|

• | /Users/macbook/Desktop/smart contracts/SHMToken.sol | dc00676d26d0e45d1a22b6edbbba2aa80e371b0c9 |

•

• Contracts Description Table

•


• | Contract | Type | Bases | | |


• |:-----:|:-----:|:-----:|:-----:|:-----:|

• | ^L | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |

• |||||




• | **Context** | Implementation | |||



• | ^L | _msgSender | Internal  | |



• | ^L | _msgData | Internal  | |



• |||||



• | **Ownable** | Implementation | Context |||

• | ^L | <Constructor> | Public   | NO  |

• | ^L | owner | Public  | | NO  |



• | ^L | renounceOwnership | Public   | onlyOwner |

• | ^L | transferOwnership | Public   | onlyOwner |




• | ^L | _setOwner | Private   | |



• |||||



• | **IERC20** | Interface | |||




• | ^L | totalSupply | External  | | NO  |

• | ^L | balanceOf | External  | | NO  |

• | ^L | transfer | External   | NO  |



• | ^L | allowance | External  | | NO  |



• | ^L | approve | External   | NO  |



• | ^L | transferFrom | External   | NO  |






















































• |||||

• | **IERC20Metadata** | Interface | IERC20 |||

• | ^L | name | External  | | NO  |

• | ^L | symbol | External  | | NO  |

• | ^L | decimals | External  | | NO  |

- |||||
- | ****ERC20**** | Implementation | Context, IERC20, IERC20Metadata |||
- | ^L | <Constructor> | Public  | NO  |
- | ^L | name | Public  | | NO  |
- | ^L | symbol | Public  | | NO  |
- | ^L | decimals | Public  | | NO  |
- | ^L | totalSupply | Public  | | NO  |
- | ^L | balanceOf | Public  | | NO  |
- | ^L | transfer | Public  |  | NO  |
- | ^L | allowance | Public  | | NO  |
- | ^L | approve | Public  |  | NO  |
- | ^L | transferFrom | Public  |  | NO  |
- | ^L | increaseAllowance | Public  |  | NO  |
- | ^L | decreaseAllowance | Public  |  | NO  |
- | ^L | _transfer | Internal  |  | |
- | ^L | _mint | Internal  |  | |
- | ^L | _burn | Internal  |  | |
- | ^L | _approve | Internal  |  | |
- | ^L | _beforeTokenTransfer | Internal  |  | |
- | ^L | _afterTokenTransfer | Internal  |  | |
- |||||
- | ****ERC20Burnable**** | Implementation | Context, ERC20 |||
- | ^L | burn | Public  |  | NO  |
- | ^L | burnFrom | Public  |  | NO  |
- |||||
- | ****SHMTToken**** | Implementation | ERC20, ERC20Burnable, Ownable |||
- | ^L | <Constructor> | Public  |  | ERC20 |
- | ^L | mint | Public  |  | 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 2 notes.