MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

SECURITY ASSESSMENT

2021

MORBEX TOKEN AUDIT REPORT

17 NOV 2021

SECURITY ASSESMENT

TABLE OF CONTENTS

SECURITY ASSESMENT
SUMMARY

This report has been prepared for Morbex to discover issues and vulnerabilities in the source code of the Morbex project as well as any contract dependencies that were not part of an officially recognizedlibrary. A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, andTestnet Deployment 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

DISCLAIMER

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at 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 us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: 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 MotechAudit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (MotechAudit) owe no duty of care towards you or any other person, nor does MotechAudit 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 MotechAudit 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, MotechAudit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against MotechAudit, 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.

SECURITY ASSESMENT

BACKGROUND

MotechAudit was commissioned by Morbex to perform an audit of smart contracts:
https://bscscan.com/address/0x0352B52f4DDEa5a4A25173796aDf8a00DE1dc5BD
The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

SECURITY ASSESMENT

AUDIT DETAILS

AUDITED PROJECT

Morbex

DEPLOYER ADDRESS

0x5E07Fd65EdECa33C7c27a16a2393B4a9785CcCAC

CLIENT CONTACTS:

Morbex team

BLOCKCHAIN

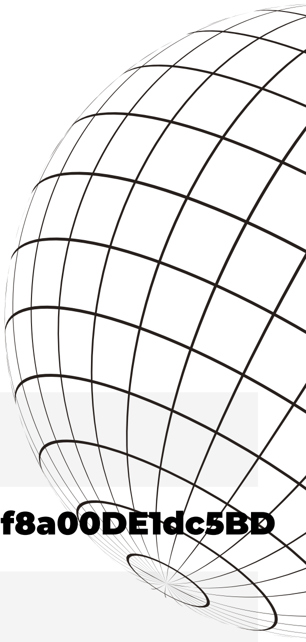
Binance smart chain Project

WEBSITE:

https://morbex.finance/

SECURITY ASSESMENT

CONTRACT DETAILS

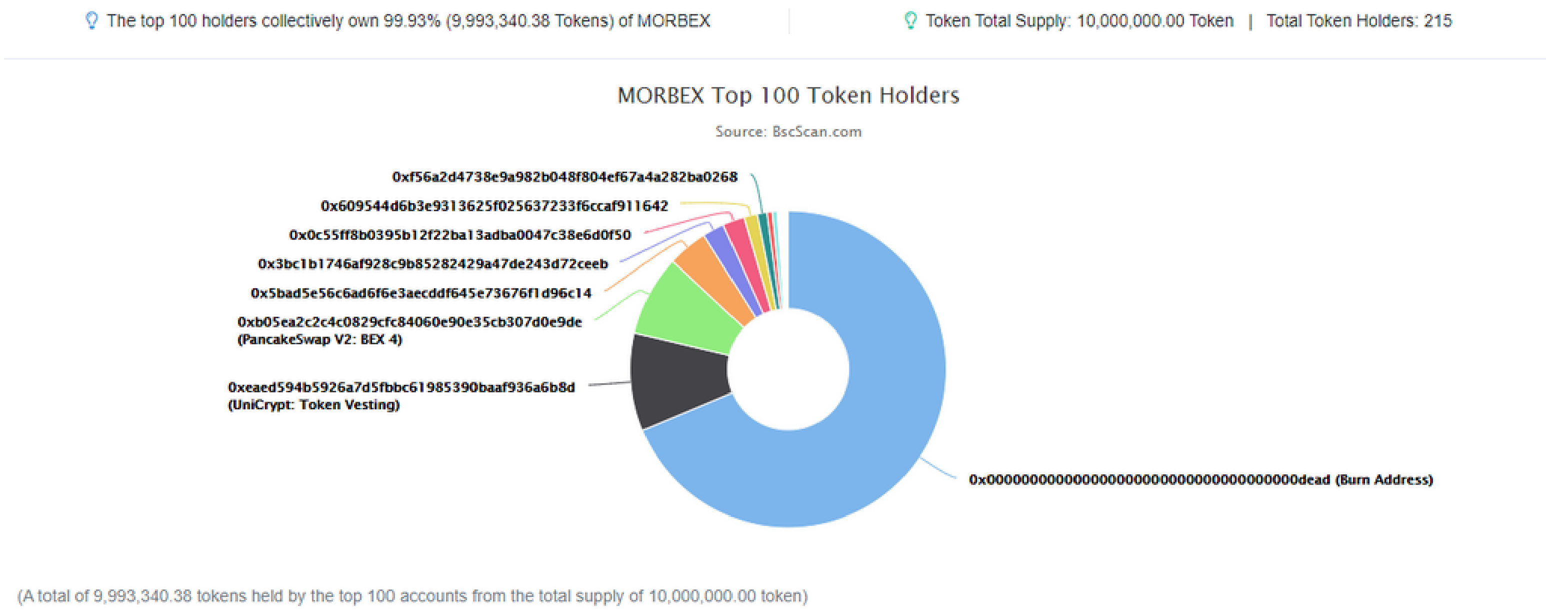


Token contract details for May-21-2021

Contract name	**Morbex**
Contract address	**0x0352B52f4DDEa5a4A25173796aDf8a00DE1dc5BD**
Total supply	**10,000,000 BEX**
Token ticker	**MORBEX (BEX)**
Decimals	**18**
Token holders	**215**
Transactions count	**1,136**
Top 100 holders dominance	**99.7851%**
Contract deployer address	**0x5E07Fd65EdECa33C7c27a16a2393B4a9785CcCAC**
Contract’s current owner address	**0x5E07Fd65EdECa33C7c27a16a2393B4a9785CcCAC**

SECURITY ASSESMENT

MORBEX TOKEN DISTRIBUTION



The top 100 holders collectively own 99.93% (9,993,340.38 Tokens) of MORBEX

Token Total Supply: 10,000,000.00 Token | Total Token Holders: 215

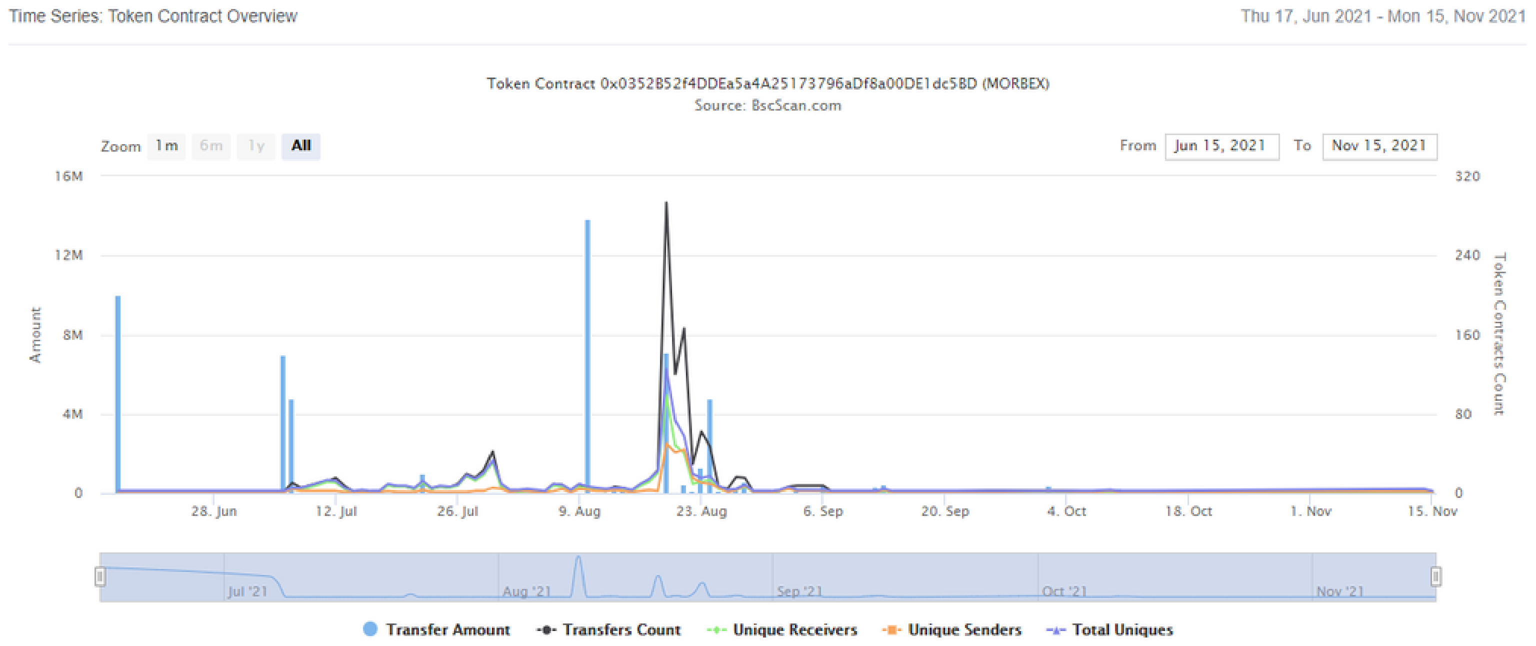
MORBEX Top 100 Token Holders

Source: BscScan.com

0xf56a2d4738e9a982b048f804ef67a4a282ba0268
0x609544d6b3e9313625f025637233f6ccaf911642
0x0c55ff8b0395b12f22ba13adba0047c38e6d0f50
0x3bc1b1746af928c9b85282429a47de243d72ceeb
0x5bad5e56c6ad6f6e3aecddf645e73676f1d96c14
0xb05ea2c2c4c0829cfc84060e90e35cb307d0e9de (PancakeSwap V2: BEX 4)
0xeaed594b5926a7d5fbbc61985390baaf936a6b8d (UniCrypt: Token Vesting)
0x000000000000000000000000000000000000dead (Burn Address)

(A total of 9,993,340.38 tokens held by the top 100 accounts from the total supply of 10,000,000.00 token)

MORBEX TOKEN CONTRACT INTERACTION DETAILS



Time Series: Token Contract Overview

Thu 17, Jun 2021 - Mon 15, Nov 2021

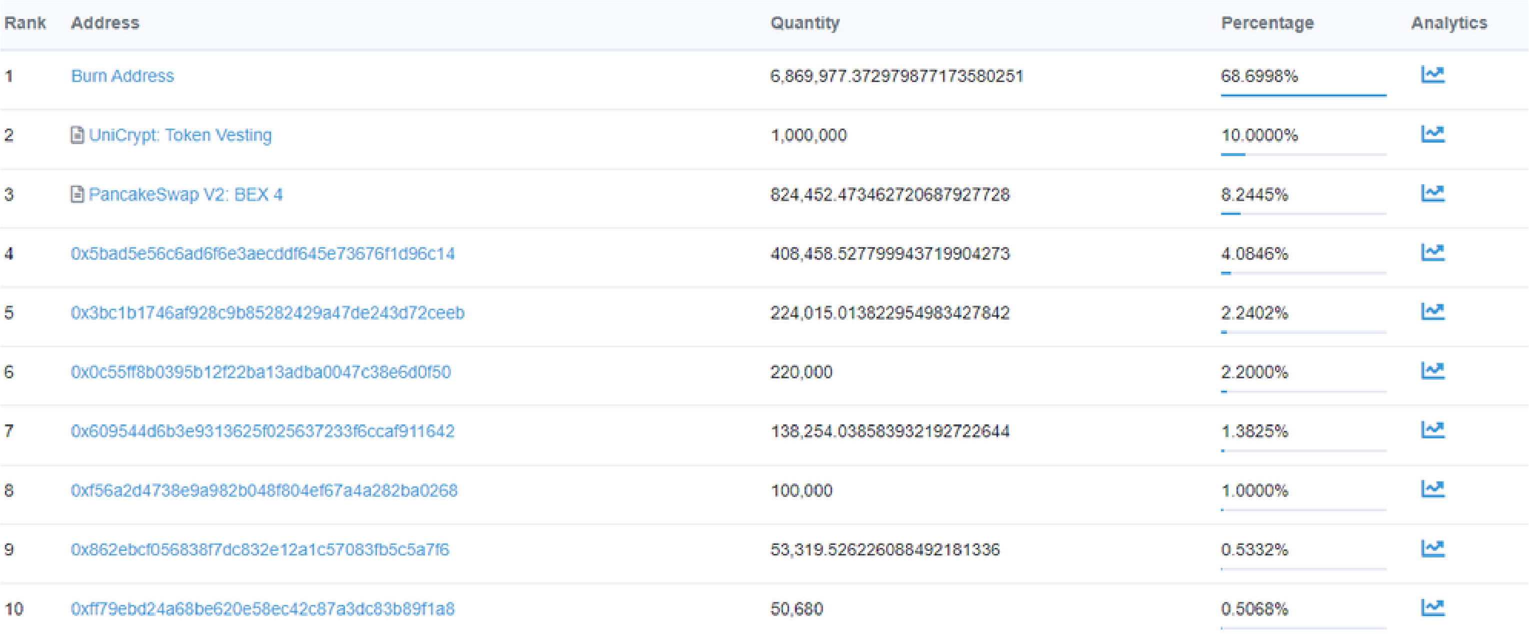
Token Contract 0x0352B52f4DDEa5a4A25173796aDf8a00DE1dc5BD (MORBEX)

Source: BscScan.com

MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

TOP 10 TOKEN HOLDERS



Rank	Address	Quantity	Percentage	Analytics
1	Burn Address	6,869,977.372979877173580251	68.6998%	📈
2	UniCrypt: Token Vesting	1,000,000	10.0000%	📈
3	PancakeSwap V2: BEX 4	824,452.473462720687927728	8.2445%	📈
4	0x5bad5e56c6ad6f6e3aecddf645e73676f1d96c14	408,458.527799943719904273	4.0846%	📈
5	0x3bc1b1746af928c9b85282429a47de243d72ceeb	224,015.013822954983427842	2.2402%	📈
6	0x0c55ff8b0395b12f22ba13adba0047c38e6d0f50	220,000	2.2000%	📈
7	0x609544d6b3e9313625f025637233f6ccaf911642	138,254.038583932192722644	1.3825%	📈
8	0xf56a2d4738e9a982b048f804ef67a4a282ba0268	100,000	1.0000%	📈
9	0x862ebcf056838f7dc832e12a1c57083fb5c5a7f6	53,319.526226088492181336	0.5332%	📈
10	0xff79ebd24a68be620e58ec42c87a3dc83b89f1a8	50,680	0.5068%	📈

source:https://bscscan.com/

SECURITY ASSESMENT

SECURITY ISSUES

✓ High Severity Issues

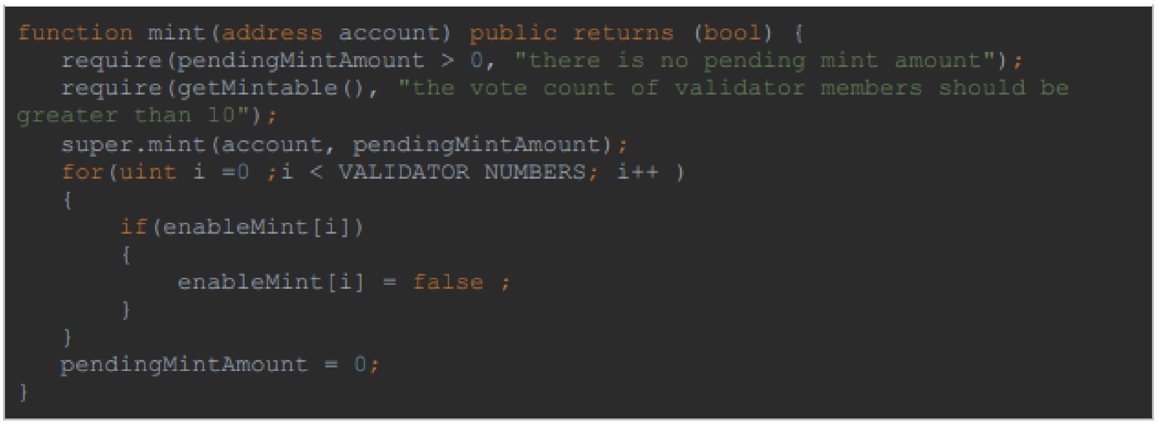
No high severity issues found.

Medium Severity Issues

1. Vulnerability: No return statement.

The mint function declared to return a boolean value but doesn't have a return statement in the body. That means the function will always return false which could be wrongly interpreted by the caller.

Lines: #310-322



```
function mint(address account) public returns (bool) {
    require(pendingMintAmount > 0, "there is no pending mint amount");
    require(getMintable(), "the vote count of validator members should be
greater than 10");
    super.mint(account, pendingMintAmount);
    for(uint i =0 ;i < VALIDATOR NUMBERS; i++ )
    {
        if(enableMint[i])
        {
            enableMint[i] = false ;
        }
    }
    pendingMintAmount = 0;
}
```

Low Severity Issues

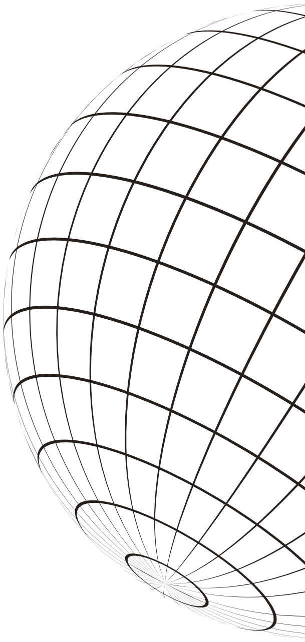
1.Vulnerability: Costly loops

Instead of building logic on loops, which is costly in the mean of gas, it's better to design the logic on the state and math.

For example, instead of looping through all validators to find a number of approves it's better to just keep this number in the state variable and update on the voting and minting request.

The same works for other places, so there's no need in loops at all.

Lowest / Code style / Best Practice Severity Issues



1. Vulnerability: Code layout.

Solidity declares the code layout recommendations that should be followed. Such recommendations include:

- Indentation
- Blank Lines
- Maximum Line Length
- Order of Functions
- Whitespace in Expressions
- Control Structures
- Function Declaration.

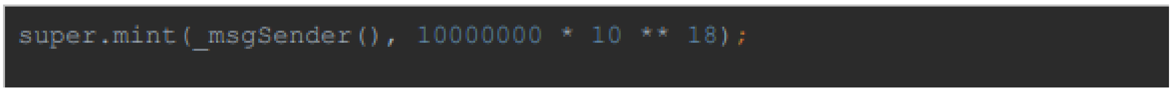
Recommendation: Please follow code layout recommendations.

2. Vulnerability: Too many digits.

Literals with many digits are difficult to read and review.

Recommendation: Please use ether units suffixes and scientific notation. Ex.: 10e6 ether

Lines: #229

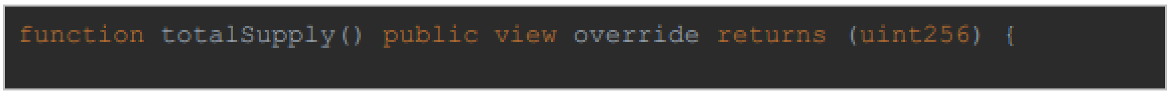


```
super.mint(_msgSender(), 10000000 * 10 ** 18);
```

3. Vulnerability: Public function that could be declared external.

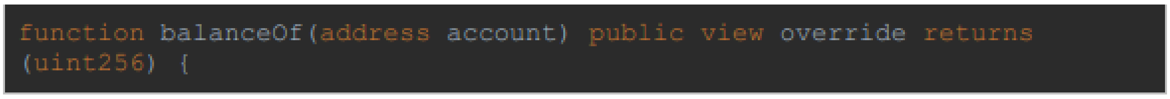
public functions that are never called by the contract should be declared external to save gas.

Lines: #75



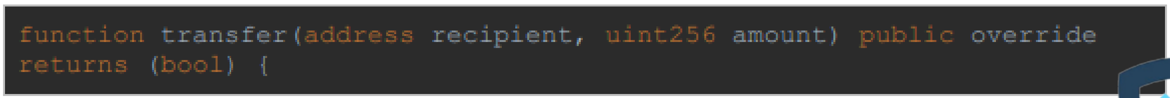
```
function totalSupply() public view override returns (uint256) {
```

Lines: #78



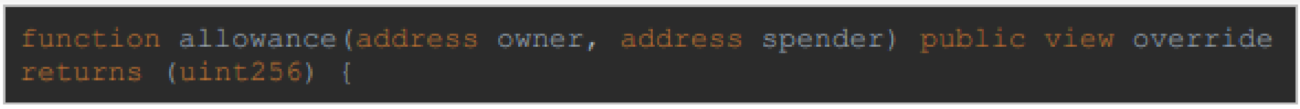
```
function balanceOf(address account) public view override returns
(uint256) {
```

Lines: #81



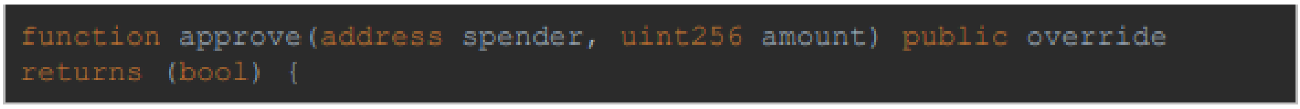
```
function transfer(address recipient, uint256 amount) public override
returns (bool) {
```

Lines: #85



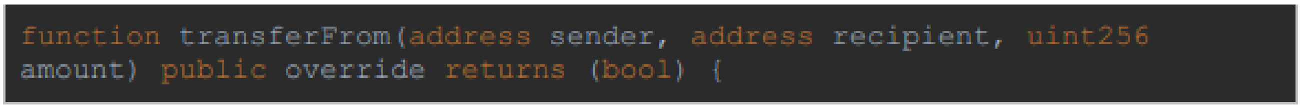
```
function allowance(address owner, address spender) public view override
returns (uint256) {
```

Lines: #88



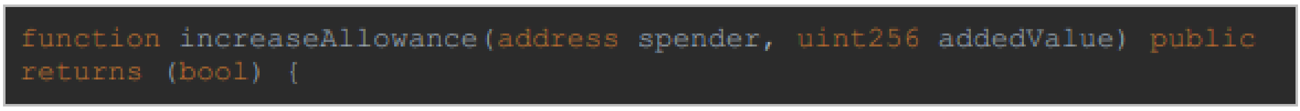
```
function approve(address spender, uint256 amount) public override
returns (bool) {
```

Lines: #92



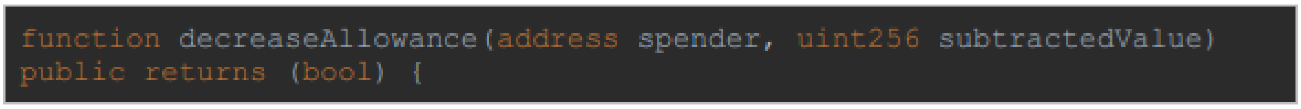
```
function transferFrom(address sender, address recipient, uint256
amount) public override returns (bool) {
```

Lines: #97



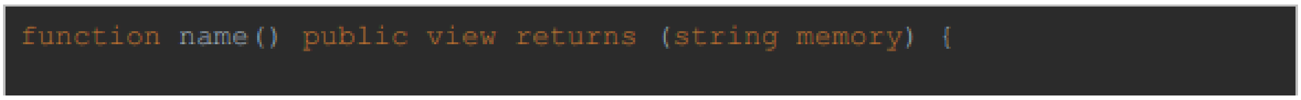
```
function increaseAllowance(address spender, uint256 addedValue) public
returns (bool) {
```

Lines: #101



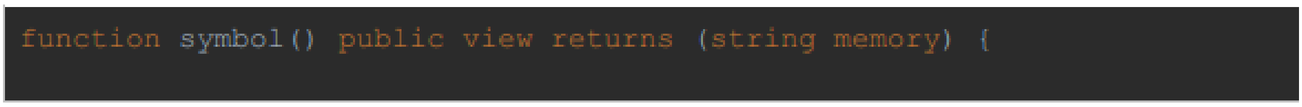
```
function decreaseAllowance(address spender, uint256 subtractedValue)
public returns (bool) {
```

Lines: #145



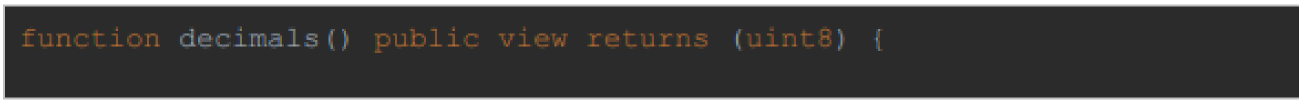
```
function name() public view returns (string memory) {
```

Lines: #148



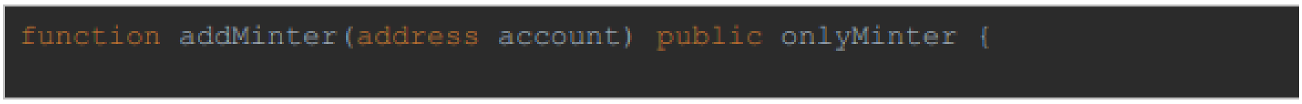
```
function symbol() public view returns (string memory) {
```

Lines: #151



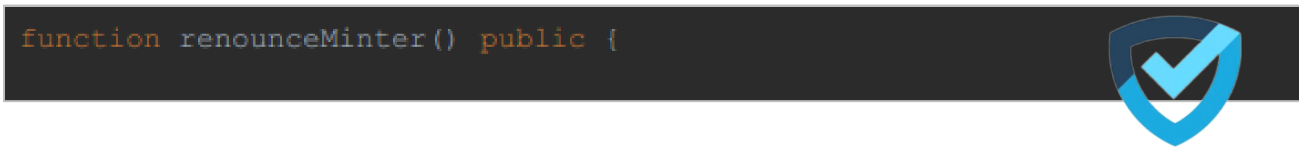
```
function decimals() public view returns (uint8) {
```

Lines: #189



```
function addMinter(address account) public onlyMinter {
```

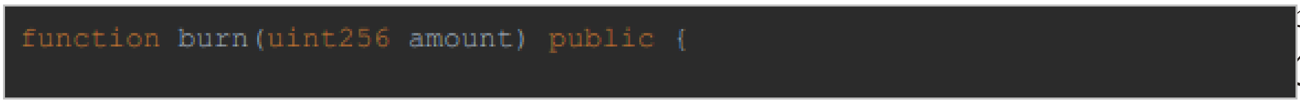
Lines: #192



```
function renounceMinter() public {
```

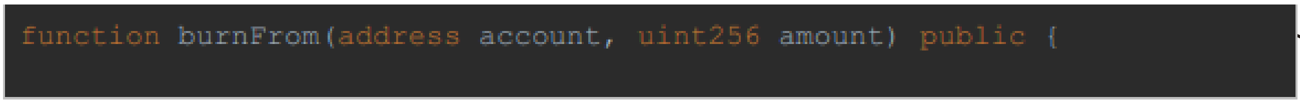
Lines: #213

Lines: #213



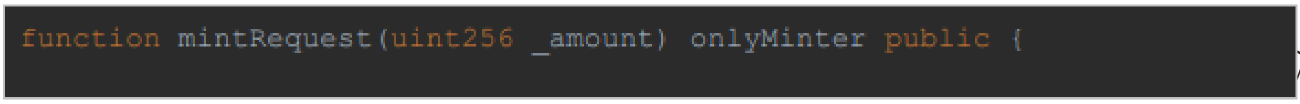
```
function burn(uint256 amount) public {
```

Lines: #216



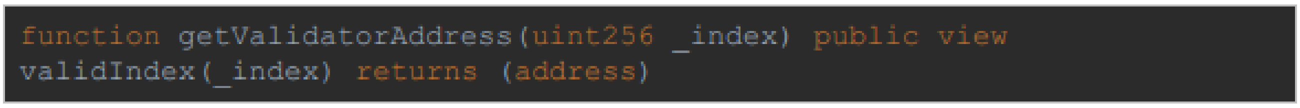
```
function burnFrom(address account, uint256 amount) public {
```

Lines: #235



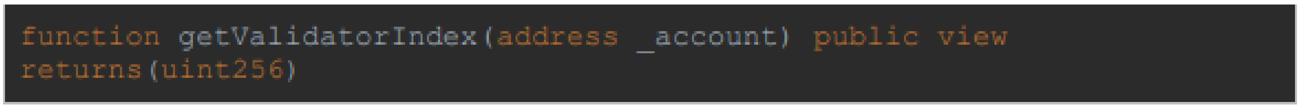
```
function mintRequest(uint256 _amount) onlyMinter public {
```

Lines: #258



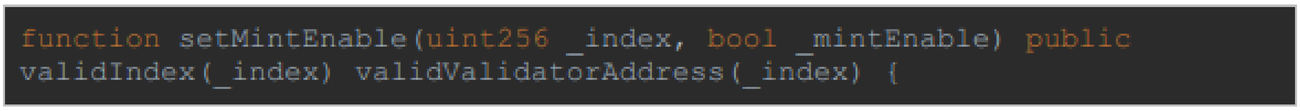
```
function getValidatorAddress(uint256 _index) public view
validIndex(_index) returns (address)
```

Lines: #262



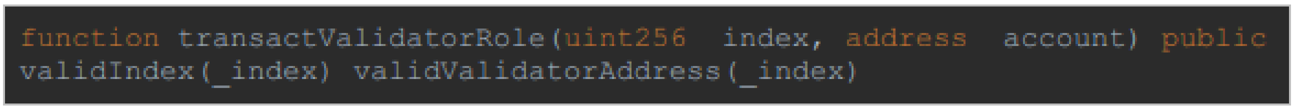
```
function getValidatorIndex(address _account) public view
returns(uint256)
```

Lines: #273



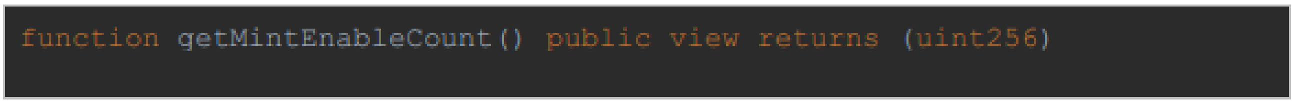
```
function setMintEnable(uint256 _index, bool _mintEnable) public
validIndex(_index) validValidatorAddress(_index) {
```

Lines: #276



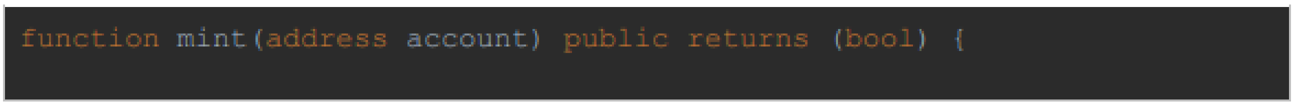
```
function transactValidatorRole(uint256  index, address  account) public
validIndex(_index) validValidatorAddress(_index)
```

Lines: #298



```
function getMintEnableCount() public view returns (uint256)
```

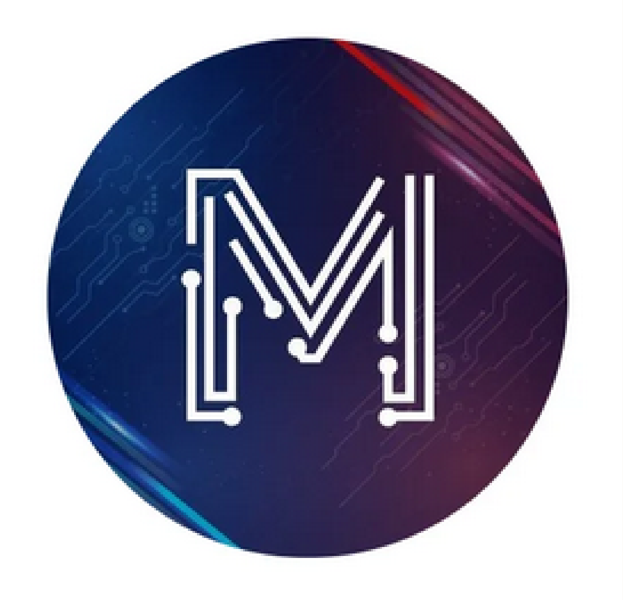
Lines: #310

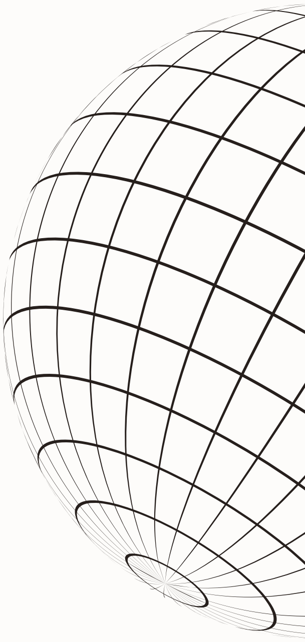


```
function mint(address account) public returns (bool) {
```

SECURITY ASSESMENT

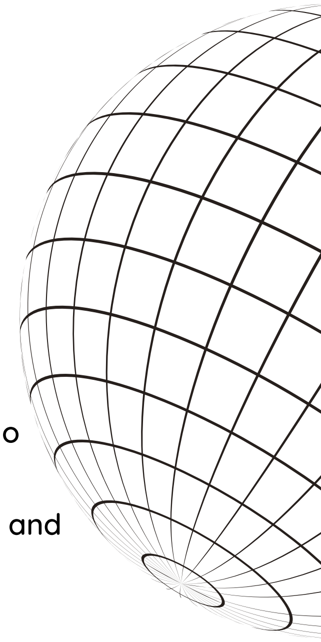
TOKEN LOGO





SECURITY ASSESMENT

CONCLUSION



Smart contracts contain owner privileges!

Motech Audit note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.