



Cyberscope

Audit Report

Meta Launcher

March 2023

Github <https://github.com/MetaLauncher/MTLA>
Commit 9fa5b376e4c80f703148f415279152a06d17d641
Audited by © cyberscope

Table of Contents

| | |
|---|-----------|
| Table of Contents | 1 |
| Review | 1 |
| Audit Updates | 2 |
| Source Files | 2 |
| Analysis | 2 |
| ST - Stops Transactions | 3 |
| Description | 3 |
| Recommendation | 0 |
| BC - Blacklists Addresses | 4 |
| Description | 4 |
| Recommendation | 5 |
| Diagnostics | 5 |
| IDI - Immutable Declaration Improvement | 6 |
| Description | 7 |
| Recommendation | 7 |
| L09 - Dead Code Elimination | 7 |
| Description | 7 |
| Recommendation | 8 |
| Functions Analysis | 8 |
| Inheritance Graph | 10 |
| Flow Graph | 10 |
| Summary | 10 |
| Disclaimer | 13 |
| About Cyberscope | 14 |

Review

| | |
|-----------------------|---|
| Contract Name | BEP20 |
| Repository | https://github.com/MetaLauncher/MTLA |
| Commit | 9fa5b376e4c80f703148f415279152a06d17d641 |
| Testing Deploy | https://testnet.bscscan.com/address/0x8499b3589f39702be1c6a0713775bfbef21c1123 |
| Symbol | MTLA |
| Decimals | 18 |
| Total Supply | 500,000,000 |

Audit Updates

| | |
|----------------------|-------------|
| Initial Audit | 04 Mar 2023 |
|----------------------|-------------|

Source Files

| | |
|---|--|
| Filename | SHA256 |
| contracts/testingDeploy/MTLA.sol | 1010279b24eb84b87eac0404a5040846a990cfe4a330d17021ac8723af6e3968 |

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

| Severity | Code | Description | Status |
|----------|------|------------------------------------|------------|
| ● | ST | Stops Transactions | Unresolved |
| ● | OCTD | Transfers Contract's Tokens | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | ULTW | Transfers Liquidity to Team Wallet | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Unresolved |

ST - Stops Transactions

| | |
|--------------------|---------------------------------------|
| Criticality | Critical |
| Location | contracts/testingDeploy/MTLA.sol#L256 |
| Status | Unresolved |

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `blockForPenaltyEnd` to a high value. As a result, the contract will blacklist all the buyers and operate as a honeypot.

```
if(earlyBuyPenaltyInEffect() ){
    if(!boughtEarly[to]){
        boughtEarly[to] = true;
        botsCaught += 1;
        emit CaughtEarlyBuyer(to);
    }
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

| | |
|--------------------|---------------------------------------|
| Criticality | Critical |
| Location | contracts/testingDeploy/MTLA.sol#L192 |
| Status | Unresolved |

Description

The contract owner has the authority to massively stop addresses from transactions. The owner may take advantage of it by calling the `massManageBoughtEarly` function.

```
function massManageBoughtEarly(address[] calldata wallets, bool flag) external  
onlyOwner {  
    for(uint256 i = 0; i < wallets.length; i++){  
        boughtEarly[wallets[i]] = flag;  
    }  
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|----------|------|-----------------------------------|------------|
| ● | IDI | Immutable Declaration Improvement | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |

IDI - Immutable Declaration Improvement

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/MTLA.sol#L151,152 |
| Status | Unresolved |

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
_nam  
_symbo
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L09 - Dead Code Elimination

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/MTLA.sol#L274,285,305 |
| Status | Unresolved |

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "BEP20: 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);
}

...
```

Recommendation

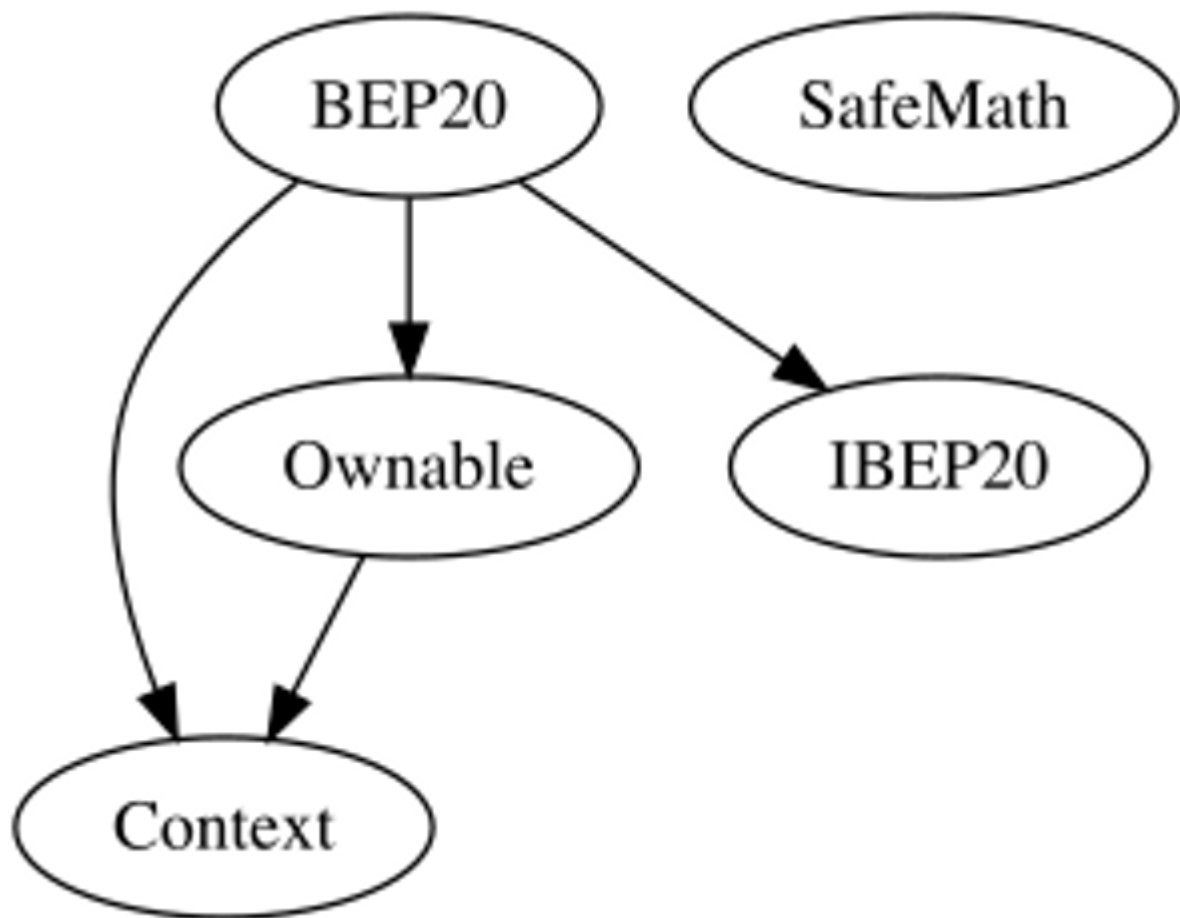
To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

| Contract | Type | Bases | | |
|-----------------|--------------------|------------|------------|-----------|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| Context | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| Ownable | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |
| | | | | |
| IBEP20 | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| SafeMath | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |

| | | | | |
|--------------|-------------------------|--------------------------------|---|-----------|
| | div | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | mod | Internal | | |
| | | | | |
| BEP20 | Implementation | Context, IBEP20, Ownable | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | enableTrading | External | ✓ | onlyOwner |
| | earlyBuyPenaltyInEffect | Public | | - |
| | manageBoughtEarly | External | ✓ | onlyOwner |
| | massManageBoughtEarly | External | ✓ | onlyOwner |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _setupDecimals | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |

Inheritance Graph



Flow Graph



Summary

Meta Launcher contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions and massively blacklist addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does 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. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope'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. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>