Audit Report

Meta Launcher

March 2023

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

Table of Contents

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

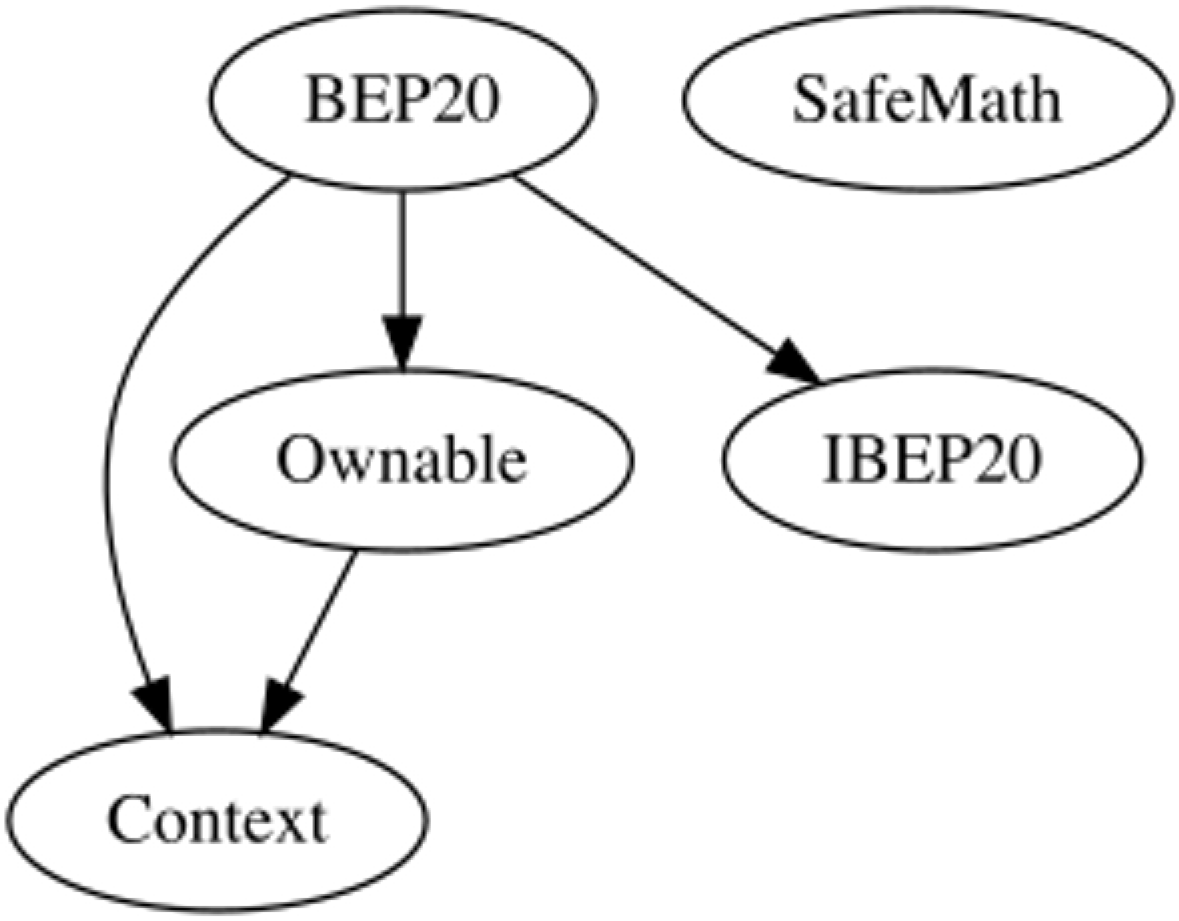
To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

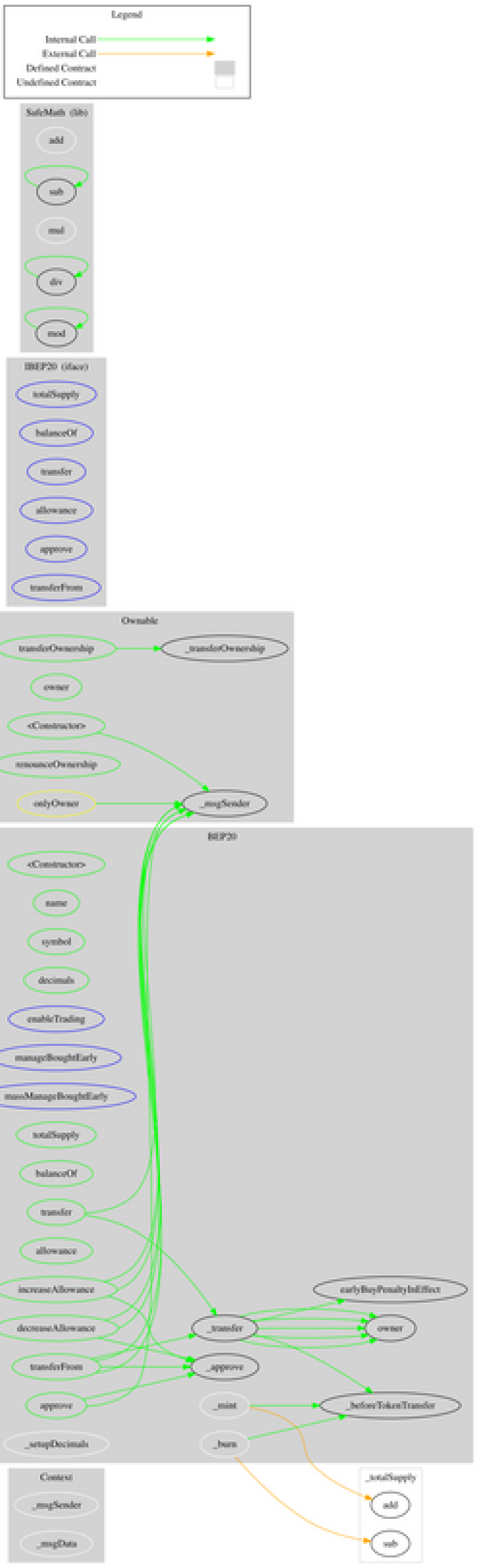
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	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


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
add
sub
mul
div
mod
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
BEP20
<Constructor>
name
symbol
decimals
enableTrading
manageBoughtEarly
massManageBoughtEarly
totalSupply
balanceOf
transfer
allowance
increaseAllowance
decreaseAllowance
transferFrom
approve
_setupDecimals
earlyBuyPenaltyInEffect
_transfer
owner
_approve
_mint
_beforeTokenTransfer
_burn
Context
_msgSender
_msgData
_totalSupply
add
sub

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

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