Cyberscope

Audit Report

Mythic Ore

November 2022

SHA256 37c72193c0c9a6e693b550bdb14b796cc35db96bbac0ebeb37bc1a5ce0d9fbaf

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	MORE
Compiler Version	v0.8.15+commit.e14f2714
Optimization	200 runs
Explorer	https://testnet.bscscan.com/token/0x6e07cD3869849227b2e60218ed96b27eF13dE76e
Symbol	MORE
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	24th November 2022
Corrected	

Source Files

Filename	SHA256
Contracts/More.sol	37c72193c0c9a6e693b550bdb14b796cc35db96bbac0ebeb37bc1a5ce0d9fbaf
Interfaces/IAgent.sol	34af3e8b8c7d60d00bf570c7161d34e7e5d95b0b385a7c2229912cb027a6e07e
Interfaces/IERC20.sol	9d801c10670382561356667549 3b12a043ed82062239ea05aa09be70b125775b
Interfaces/IUniswap.sol	0103ddebd8029270be84ca37b5b40f7512b7f9b36e86839686bdf5bedb8ad586
Libraries/LibraryListAddress.sol	862e87217386e566842721b325e10d2c7912d439f2593073c758188ec7657b8f

Contract Architecture

The contract implements an ERC20 token enriched with some features like reflections and autogenerated liquidity pool. The implementation of the contract is custom and it is not based on any well-known implementation. As a result, some concepts and methodologies like allowance, reflections, gas optimization etc. could be more well-structued. The team is adviced to fork a well-known ERC20 implementation that contains the same features and apply their requirements.

Contract Analysis

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	OTUT	Transfers User's Tokens	Unresolved
●	TSD	Total Supply Diversion	Unresolved
●	RLS	Redundant Liquidity Swaps	Unresolved
●	ZD	Zero Division	Unresolved
●	RM	Reflection Mechanism	Unresolved
●	RI	Redundant Iterations	Unresolved
●	SUV	Solidity Uncheck Vulnerabilities	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

OTUT - Transfers User's Tokens

Criticality	critical
Location	contract.sol#L238,255
Status	Unresolved

Description

The contract is not subtracting the transfer amount from allowances. If an address has approved allowance for a limited amount of tokens, then the approved allowance address can transfer all the tokens.

```
function transferFrom(address from, address to, uint256 value) external
override returns(bool)
{
    require(value > 0 && __allowances[from][msg.sender] >= value,
"transferFrom0");

    return handleTransfer(from, to, value);
}
```

```
function lightningTransfer(address sender, uint256 amount) external
onlyAuthorized()
{
    uint32 senderReflectionsMultiplier =
Modifiers[sender].reflectionsMultiplier;
    if (senderReflectionsMultiplier > 0)
    {
        updateReflections(sender);
    }

    require(__allowances[sender][msg.sender] >= amount && __balanceOf[sender]
>= amount, "shock");

    unchecked
    {
        __balanceOf[sender] -= amount;
        __balanceOf[msg.sender] += amount;

        emit Transfer(sender, msg.sender, amount);
    }

    if (senderReflectionsMultiplier > 0)
    {
        updateMultiplierBalances(sender);
    }
}
```

Recommendation

The contract should deduct the transfrerred amount from the allowance.

TSD - Total Supply Diversion

Criticality	critical
Location	contract.sol#L1077
Status	Unresolved

Description

The amount that is added to the total supply does not equal the amount that is added to the balances. As a result, the sum of balances is diverse from the total supply.

```
__balanceOf[account] += _reflected;
```

Recommendation

The sum of balances should always be equal to the total supply.

RLS - Redundant Liquidity Swaps

Criticality	minor / informative
Location	contract.sol#L1220,1241
Status	Unresolved

Description

In order to accumulate tokenLiquidityReserves the contract swap tokens for BNB and then swap back the proportional BNB for tokens.

```
function addLiquidityFromTokenReserves() private
{
    uint80 liquidityPotBefore = potsBNB.liquidity;
    potsBNB.liquidity = 0;

    (uint256 addedTokens, uint256 addedBNB,) =
SwapRouter.addLiquidityETH{value: liquidityPotBefore - 1}(
            address(this),
            tokenLiquidityReserves,
            0,
            0,
            address(this),
            block.timestamp
        );

    unchecked
    {
        potsBNB.liquidity = liquidityPotBefore - uint80(addedBNB);
        tokenLiquidityReserves -= addedTokens;
    }
}
```

```
function refillLiquidityTokenReserves() private
{
    unchecked
    {
        uint256 amountBNBtoBeSwapped = potsBNB.liquidity / 2;
        potsBNB.liquidity -= uint80(amountBNBtoBeSwapped);

        uint256 swappedTokens = swapBNBForTokens(amountBNBtoBeSwapped);
        tokenLiquidityReserves += swappedTokens;
    }
}
```

Recommendation

The contract could accumulate the tokens directly from the liquidity fees.

ZD - Zero Division

Criticality	critical
Location	contract.sol#L1036
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. More precisely, `_reflections.prevDelay` never gets initialized on the contract's deployment, so by default its initial value will be set to 0. As a result, the transactions will revert.

```
takeFromPot = _reflections.pot * _reflections.potPartToDistribute *
(_reflections.prevDelay - timeDifference) / _reflections.prevDelay /
DENOMINATOR;
```

Recommendation

The contract should not allow the method's execution if the variables are not properly initialized.

RM - Reflection Mechanism

Criticality	minor / informative
Location	contract.sol#L804,808
Status	Unresolved

Description

The contract uses a complicated technique to send the reflected tokens to each account. On every transfer, the sender's and the receiver's balance is updated according to the corresponding reflected amount. This process produces a large amount of gas cost proportionally to the number of transfers.

```
updateReflections(sender);
...
updateReflections(recipient);
```

Recommendation

The contract could use a simpler reflections mechanism that is based on a classic safemoon fork.

https://github.com/safemoonprotocol/Safemoon.sol/blob/main/Safemoon.sol

RI - Redundant Iterations

Criticality	minor / informative
Location	contract.sol#L722
Status	Unresolved

Description

The contract performs redundant iterations. Shareholders are added from _addMultiplier(). The autoCompound() iterates the Shareholders. If the maxIterations is greater then the Shareholders.length then it will execute the same shareholders.

```
function autoCompound(uint256 maxIterations) public {
    uint256 length = Shareholders.Array.length;
    if (length < 2) {
        return;
    }

    uint256 currentIndex = reflections.currentCompoundingIndex;
    uint256 iterations = 0;

    reflections.perShareStored = reflectionsPerShare();
    reflections.lastUpdateTime = lastTimeReflectionsApplicable();

    while (iterations < maxIterations) {
        address account = Shareholders.Array[currentIndex];

        payReflections(account, reflections.perShareStored);
        updateMultiplierBalances(account);

        unchecked {
            ++currentIndex;
            ++iterations;

            if (currentIndex == length) {
                currentIndex = 1;
            }
        }
    }
    reflections.currentCompoundingIndex = uint32(currentIndex);
}
```

Recommendation

The contract could prevent iterating over the same shareholders in the same execution context.

SUV - Solidity Uncheck Vulnerabilities

Criticality	critical
Location	contract.sol#L814
Status	Unresolved

Description

Since the calculations are running on a Solidity uncheck environment, then they are vulnerable to overflow attacks. For instance, if a user executes the transfer() with a huge amount, then many checks could overflow and produce a positive result. We state that this segment is a sample of potential vulnerabilities that can be produced from the uncheck statements.

```
__balanceOf[recipient] + amount - taxAmount <= workAmounts.maxAccount
```

Recommendation

The contract should not allow unchecked operations since it creates vulnerabilities.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L18
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
__totalSupply
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L73,88,16,18,21,143,145,69,20,51,147,72,146,75,70 contracts/Interfaces/IUniswap.sol#L37,54,36,71
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
AuthorizedContracts
PERMIT_TYPEHASH
Taxes
__decimals
MINIMUM_LIQUIDITY
__totalSupply
__allowances
SwapRouter
SwapAgent
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L460
Status	Unresolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
minGasForWorkOnSale = newValueOnSale
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L1117,958,1094
Status	Unresolved

Description

Performing divisions before multiplications may cause lose of prediction.

```
sellTokens = (_tokenPots.sell) * toSwap / totalTokens
buyTokens = (_tokenPots.buy) * toSwap / totalTokens
taxAmountScaled = taxAmount / TOKEN_POTS_DIVISOR
liquidityTokens = (_tokenPots.liquidity - 1) * toSwap / totalTokens
transferTokens = (_tokenPots.transfer) * toSwap / totalTokens
referrerTokens = (_tokenPots.referrer) * toSwap / totalTokens
reflections.perShareStored + (lastTimeReflectionsApplicable() - reflections.lastUpdateTime) *
(reflections.rate * ONE / reflections.totalBalances)
referrerAmount = taxAmountScaled * taxData.referrer / totalTax
...
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L863,908,976
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
txType
tax
referrerAmount
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
MORE	Implementation	IERC20		
	_onlyMain	Private		
	_onlyAuthorized	Private		
	_onlySwap	Private		
	_flagCheck	Private		
	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	transferFrom	External	✓	-
	transfer	External	✓	-
	lightningTransfer	External	✓	onlyAuthorized
	prepareReferralSwap	External	✓	onlySwap
	approve	External	✓	-
	setModifiers	External	✓	onlyAuthorized
	setModifiers	External	✓	onlyAuthorized
	addMultiplier	External	✓	onlyAuthorized
	setBuyTaxReduction	External	✓	onlyAuthorized
	setSellTaxReduction	External	✓	onlyAuthorized
	addTokensToLiquidityReservesFromContract	External	✓	onlyAuthorized
	addBNBToLiquidityPot	External	Payable	-
	buybackAndBurn	External	Payable	-
	buybackAndLockToLiquidity	External	Payable	-
	addAuthorized	External	✓	onlyMain
	removeAuthorized	External	✓	onlyMain
	lockLiquidityFromFees	External	✓	onlyMain
	withdrawLiquidityFromFees	External	✓	onlyMain
	toggleSellAddress	External	✓	onlyMain flagCheck
	toggleAccountTaxExclusion	External	✓	onlyMain

	toggleAccountMaxAccountRuleExclusion	External	✓	onlyMain flagCheck
	setReferralTaxReduction	External	✓	onlyMain
	setMaxAccountAndMaxMultiplier	External	✓	onlyMain
	setReflectionsDelayAndDistributingPart	External	✓	onlyMain
	setMaxCompoundingIterations	External	✓	onlyMain
	setMinGasForWork	External	✓	onlyMain
	setTax	External	✓	onlyMain
	setWorkAmounts	External	✓	onlyMain
	setMainAccount	External	✓	onlyMain
	setAgents	External	✓	onlyMain
	addToReflectionsFromContract	External	✓	onlyAuthorized
	withdrawFreeBNB	External	✓	onlyMain
	withdrawFreeTokens	External	✓	onlyMain
	launchToken	External	✓	onlyMain
	balanceOf	External		-
	rawBalanceOf	External		-
	lastReferrerTokensAmount	External		-
	getModifiers	External		-
	getModifiers	External		-
	isAuthorized	External		-
	allowance	External		-
	totalSupply	External		-
	circulatingSupply	External		-
	viewTaxes	External		-
	viewShareholders	External		-
	viewAuthorized	External		-
	decimals	External		-
	doWork	Public	✓	-
	doExcessiveWork	Private		
	autoCompound	Public	✓	-
	compoundReflections	Public	✓	-
	reflected	Public		-
	reflected	Private		
	getFreeTokens	Public		-

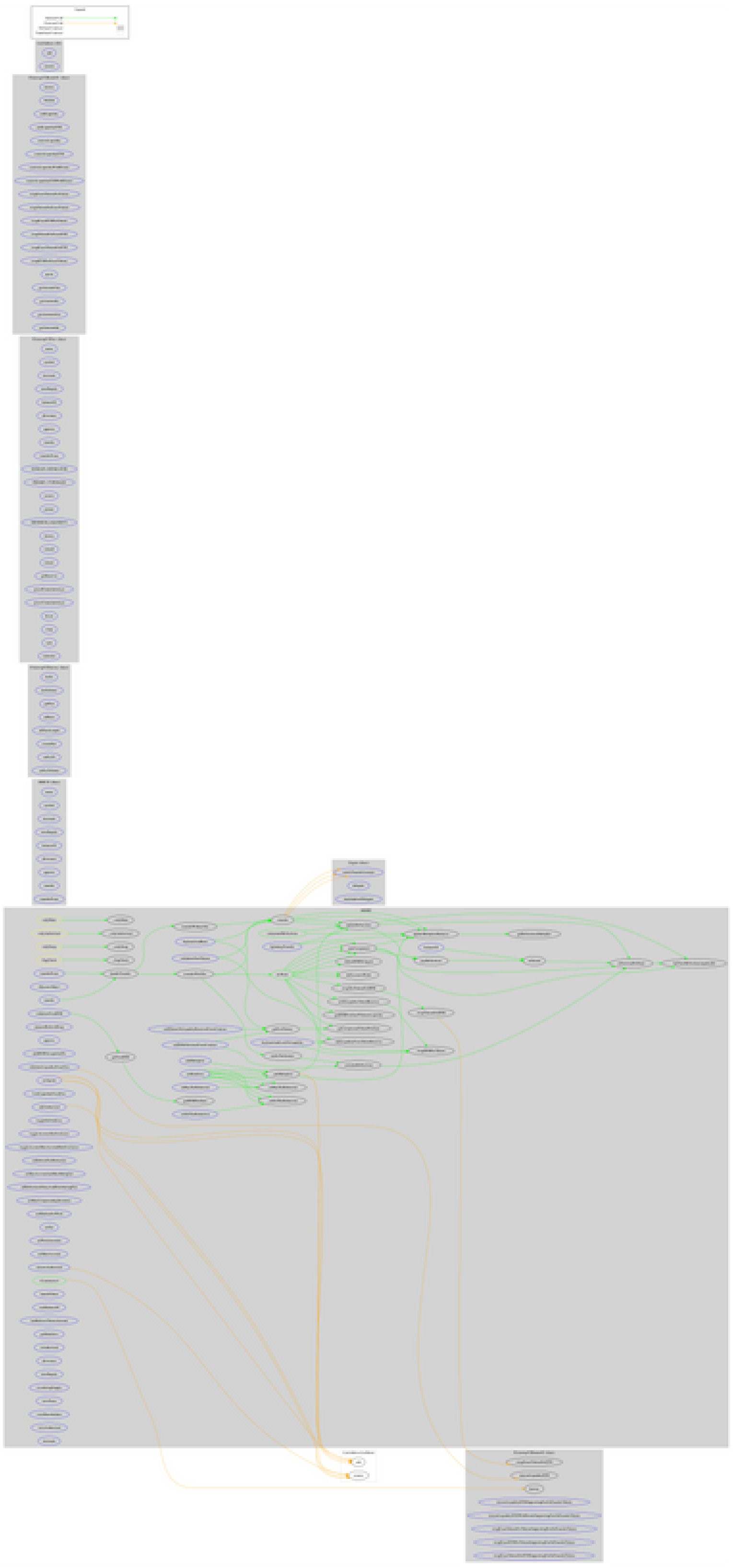
	getFreeBNB	Public		-
	_transfer	Internal	✓	
	handleTransfer	Private	✓	
	transferWithTax	Private	✓	
	transferWithoutTax	Private	✓	
	deliverBNBToAgent	Private	✓	
	notifyTaxSystem	Private	✓	
	calculateReflections	Private	✓	
	updateMultiplierBalances	Private	✓	
	updateReflections	Private	✓	
	payReflections	Private	✓	
	lastTimeReflectionsApplicable	Private		
	reflectionsPerShare	Private		
	getReflectionsMultiplier	Private		
	swapTaxTokensForBNB	Private	✓	
	swapTokensForBNB	Private	✓	
	swapBNBForTokens	Private	✓	
	addLiquidityFromTokenReserves	Private	✓	
	refillLiquidityTokenReserves	Private	✓	
	_addMultiplier	Private	✓	
	_setBuyTaxReduction	Private	✓	
	_setSellTaxReduction	Private	✓	
	getCompressedTokenPotsSum	Private		
	getBNBPotsSumWithoutLiquidity	Private		
	getBNBPotsSum	Private		
IAgent	Interface			
	delegate	External	Payable	-
	marketplaceDelegate	External	Payable	-
	notifyTransferListener	External	✓	-
	notifyTransferListener	External	✓	-
IERC20	Interface			
	name	External		-
	symbol	External		-

	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-

	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	swap	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-

	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
ListAddress	Library			
	add	External	✓	-
	remove	External	✓	-

Contract Flow



Summary

The Mythic Ore contract implements an ERC20 token. This audit investigates security issues, business logic concerns and potential improvements.

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