Cyberscope

Audit Report

Mythic Ore

November 2022

SHA256 4ba22f0d4112620cf01fd515a2eb1ad4eeed184043d553f566dab0c1580fcf83

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	MORE
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Test Deploy	https://testnet.bscscan.com/token/0xeb06d2063d6F79cd34099Ee083972Fd5fD2b4ED7
Symbol	MORE
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	24th November 2022 https://github.com/cyberscope-io/audits/tree/main/Mythic_Ore/v1/audit.pdf
Corrected Phase 1	29th November 2022 https://github.com/cyberscope-io/audits/tree/main/Mythic_Ore/v2/audit.pdf
Corrected Phase 2	30th November 2022

Source Files

Filename	SHA256
Contracts/More.sol	4ba22f0d4112620cf01fd515a2eb1ad4eeed184043d553f566dab0c1580fcf83
Interfaces/IAgent.sol	f8c9223dd1fcfbfb007033f3a858c878817ab3690c8a490670e1620a2742d3a6
Interfaces/IERC20.sol	8e4432d03fcab96a31b3325d04d921e505e435b8f1a3c771de4ec35a3af0c207
Interfaces/IUniswap.sol	791d1ec4ab5cc06068d70ee99e048b4100d3264b6b1d99d00baf82b94f3c5126
Libraries/LibraryListAddress.sol	9a2313ff0f4bbfa837208a4c10acc54db8c9c58ea910f91419469e8754ec6446

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.

Team's Reply 29 November 2022

The team has acknowledged that this is not a security issue and states:

"We are not using Safemoon-fork reflections system cause:

1. *It is expensive in terms of gas too.*
2. *It is harder to implement considering we have a different method of share calculations.*
3. *So all gas optimizations are targeted mainly to compensate costly reflections system."*

Contract Analysis

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TSD	Total Supply Diversion	Acknowledged
●	RLS	Redundant Liquidity Swaps	Acknowledged
●	RM	Reflection Mechanism	Acknowledged
●	SUV	Solidity Uncheck Vulnerabilities	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Acknowledged

TSD - Total Supply Diversion

Criticality	medium
Location	contract.sol#L1158
Status	Acknowledged

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.

Team’s Reply 29 November 2022

The team has acknowledged that this is not a security issue and states:

“This amount is deducted from the total supply in notifyTaxSystem() then its continuously being added here until reflection cycle is finished while it is true that the sum of all balances won't be equal to the total supply in most cases in the end, they will become equal (minus some small amount as a result of rounding down on divisions) but sum of all balances will not ever be greater than total supply so your comment that "The amount that is added to the total supply does not equal the amount that is added to the balances" is not true because tokens here are not actually being minted or added to the total supply but as stated in a comment above it is true that sum of all balances will be slightly lower than total supply presonally I do not think this is a major issue.”

RLS - Redundant Liquidity Swaps

Criticality	minor / informative
Location	contract.sol#L1318,1339
Status	Acknowledged

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.

Team’s Reply 29 November 2022

The team has acknowledged that this is not a security issue and states:

“Initially tokenLiquidityReserves is filled so no reduntant swaps then after some time it is true that redundant swap are gonna happen, but:

1. *it saves just a little bit of gas to not write to tokenLiquidityReserves on every taxed tx*
2. *it is not useful to write every time to tokenLiquidityReserves as initially it is filled*

we were planning to fill it after deployment but it is surely more reasonable to do it here that way instead of using 50/50 system we forward 100% bnb from tax to liquidity and it works while tokenLiquidityReserves is filled not as a result of a swap.”

RM - Reflection Mechanism

Criticality	minor / informative
Location	contract.sol#L862,866
Status	Acknowledged

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

Team's Reply 29 November 2022

The team has acknowledged that this is not a security issue.

The team response is mentioned in the Contract Architecture section.

SUV - Solidity Uncheck Vulnerabilities

Criticality	critical
Location	contract.sol#L302-1357
Status	Acknowledged

Description

The unchecked statements in large-scale usage are not a good practice in Solidity. They are a source of vulnerabilities and attacks. They should be used in a small and controlled environment. The potential exploits are increased logarithmically related to the unchecked code size.

```
unchecked
{
    uint256 agentPotToSend = getBNBPotsSumWithoutLiquidity();
    if (gasleft() > minGasForWorkOnSale)
    ...
}
...
...
unchecked
{
    ++currentIndex;
    ++iterations;
    ...
}
...
unchecked
{
    return Modifiers[account].oldMultiplierBalance * (reflectionsPerShare() -
ReflectionsPerSharePaid[account]) / ONE;
}
...
unchecked
{
    return Modifiers[account].oldMultiplierBalance * (perShareStored -
ReflectionsPerSharePaid[account]) / ONE;
}
...
unchecked
```

```
{
    return _balanceOf[address(this)] - getCompressedTokenPotsSum() *
TOKEN_POTS_DIVISOR - tokenLiquidityReserves;
}
...
unchecked
{
    if (txType == 0)
    {
        taxReduction += Modifiers[recipient].buyTaxReduction;
    }
    ...
}
...
unchecked
{
    uint256 taxAmountScaled = taxAmount / TOKEN_POTS_DIVISOR;
    uint256 reflectionsAmount = taxAmountScaled * taxData.reflections /
totalTax;
    ...
}
...
unchecked
{
    ReflectionsData storage _reflections = reflections;
    _reflections.pot += uint96(reflectionsAmount);
    _reflections.perShareStored = reflectionsPerShare();
    ...
}
...
unchecked
{
    uint256 usingMultiplier = getReflectionsMultiplier(account);
    uint256 newBalance = _balanceOf[account] * usingMultiplier / ONE / 100;
    reflections.totalBalances = uint64(reflections.totalBalances -
Modifiers[account].oldMultiplierBalance + newBalance);
    Modifiers[account].oldMultiplierBalance = uint64(newBalance);
}
...
unchecked
{
    PotsDataToken storage _tokenPots = tokenPots;
    uint256 toSwap;
    uint256 liquidityTokens;
    uint256 buyTokens;
    ...
}
...
unchecked
{
```

```
    potsBNB.liquidity = liquidityPotBefore - uint80(addedBNB);
    tokenLiquidityReserves -= addedTokens;
}
...
unchecked
{
    uint256 amountBNBtoBeSwapped = potsBNB.liquidity / 2;
    potsBNB.liquidity -= uint80(amountBNBtoBeSwapped);
    uint256 swappedTokens = swapBNBForTokens(amountBNBtoBeSwapped);
    tokenLiquidityReserves += swappedTokens;
}
...
unchecked
{
    if (isAddition == 1)
    {
        uint32 newMultiplier = Modifiers[account].reflectionsMultiplier +
difference;
    ...
}
```

Recommendation

The team is advised to remove the unchecked statements. If it is required for gas optimization then it should be limited to small segments. One example is decreasing the balance from accounts after ensuring the amount with require() statement.

References

- https://nvd.nist.gov/vuln/detail/CVE-2018-10299
- https://github.com/sigp/solidity-security-blog#2-arithmetic-overunder-flows-1
- https://dreamlab.net/en/blog/post/ethereum-smart-contracts-vulnerabilities-integer-overflow-and-underflow/

Team’s Reply 29 November 2022

“Removing unchecked from this block of code fixes the issue in the audit it is marked on the line `_balanceOf[recipient] + amount - taxAmount <= workAmounts.maxAccount` *and while the check above can overflow, then it will result in a reverted tx here as it is impossible to have an amount of tokens on balance that can result in the overflow.”*

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/Contracts/More.sol#L81,24,96,151,153,77,59,155,80,20,154,83
Status	Acknowledged

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
_totalSupply
Taxes
SwapRouter
SwapAgent
ReflectionsPerSharePaid
Modifiers
Agent
Shareholders
...
```

Recommendation

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

Team Update 29 November 2022

The team has acknowledged that this is not a security issue.

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		-

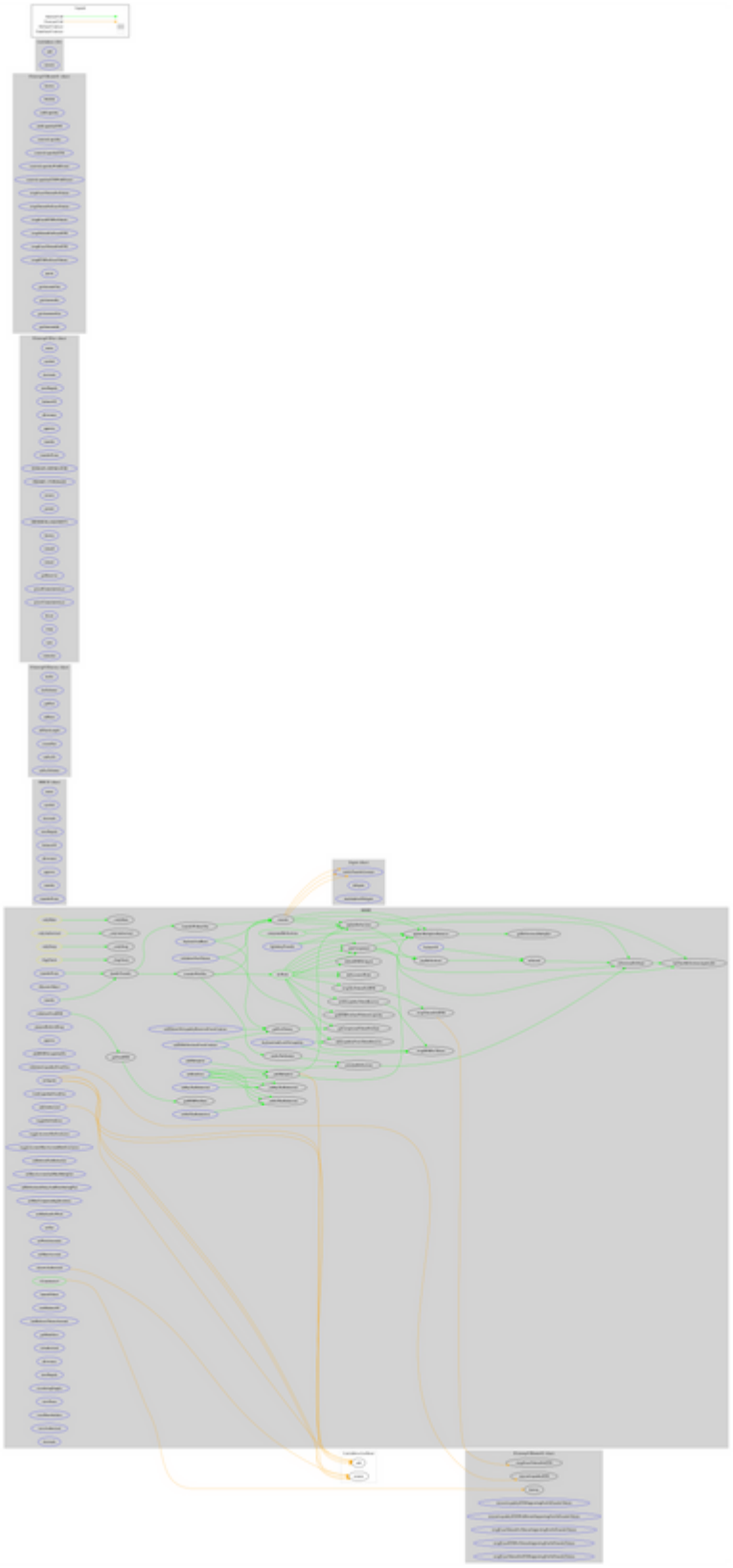
	getFreeBNB	Public		-
	_transfer	Internal	✓	
	handleTransfer	Private	✓	
	transferWithTax	Private	✓	
	transferWithoutTax	Private	✓	
	deliverBNBToAgent	Private	✓	
	notifyTaxSystem	Private	✓	
	calculateReflections	Private	✓	
	updateMultiplierBalances	Private	✓	
	updateReflections	Private	✓	
	payReflections	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	IUniswapV2 Router01		
	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.

Team Update 29 November 2022

The team has replied to all of the findings and has acknowledged that the remaining are not security issues.

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