Cyberscope

Audit Report

Meme Casino

November 2023

Network   ETH

Address   0x6ffdb458c324687cdc03bb1305092c211e781147

Audited by   © cyberscope

Analysis

● Critical   ● Medium   ● Minor / Informative   ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical   ● Medium   ● Minor / Informative

Severity	Code	Description	Status
●	ROI	Renounce Ownership Inconsistency	Unresolved
●	MC	Missing Check	Unresolved
●	RSW	Redundant Storage Writes	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
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Explorer	https://etherscan.io/address/0x6ffdb458c324687cdc03bb13050 92c211e781147
Symbol	MEMEC
Decimals	18
Total Supply	1,000,000

Audit Updates
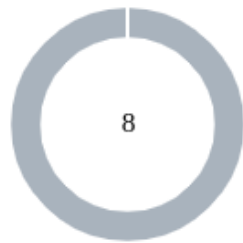
Initial Audit	27 Oct 2023 https://github.com/cyberscope-io/audits/blob/main/memec/v1/a udit.pdf
Corrected Phase 2	01 Nov https://github.com/cyberscope-io/audits/blob/main/memec/v2/a udit.pdf
Corrected Phase 3	03 Nov 2023

Source Files

Filename	SHA256
MemeCasino.sol	8731d42e9e58a4502ecd2690d3a5c15bff8305a151d229635ade6114e9 9c0690

Findings Breakdown

	Critical	0
	Medium	0
	Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	8	0	0	0

ROI - Renounce Ownership Inconsistency

Criticality	Minor / Informative
Location	MemeCasino.sol#L233
Status	Unresolved

Description

The contract uses the `renounceOwnership` function which transfers the ownership of the contract to the zero address, effectively renouncing the possibility of future ownership. This function includes a `require` statement that checks the state of the `_hasLiqBeenAdded` boolean variable, expecting it to be false in order to proceed with the renouncement. However, there is a discrepancy between this logic and the associated comment, which implies that ownership should only be renounced after trading has been enabled, a state indicated by `_hasLiqBeenAdded` being true. As a result, once liquidity has been added, the `renounceOwnership` function becomes unusable, as the require condition can never be met.

```
function renounceOwnership() external onlyOwner {
    require(!_hasLiqBeenAdded, "Cannot renounce until
trading has been enabled.");
        ...
    emit OwnershipTransferred(oldOwner, address(0));
  }
```

Recommendation

It is recommended to align the `renounceOwnership` function's logic with the intended behavior as suggested by the comment. If the purpose is indeed to allow the renouncement of ownership after the parameter has been enabled, the `require` statement should be modified to assert `_hasLiqBeenAdded` is true. The code should be updated to `require(_hasLiqBeenAdded,)` to reflect this intention. Additionally, the comment should be clarified to accurately describe the condition under which `renounceOwnership` can be successfully called. This change will ensure that the function behaves as expected and that the contract's documentation is consistent with its actual functionality.

MC - Missing Check

Criticality	Minor / Informative
Location	MemeCasino.sol#L379
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

Specifically the parameters `thresholdDivisor` and `amountDivisor` can be set to zero.

```
 function setSwapSettings(uint256 thresholdPercent, uint256
thresholdDivisor, uint256 amountPercent, uint256 amountDivisor)
external onlyOwner {
        swapThreshold = (_tTotal * thresholdPercent) /
thresholdDivisor;
        swapAmount = (_tTotal * amountPercent) / amountDivisor;
        ...
    }
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	MemeCasino.sol#L369
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```solidity
    function setExcludedFromLimits(address account, bool
enabled) external onlyOwner {
        _isExcludedFromLimits[account] = enabled;
    }

    function setExcludedFromFees(address account, bool enabled)
public onlyOwner {
        _isExcludedFromFees[account] = enabled;
    }

    function setExcludedFromProtection(address account, bool
enabled) external onlyOwner {
        _isExcludedFromProtection[account] = enabled;
    }

    function removeSniper(address account) external onlyOwner {
        initializer.removeSniper(account);
    }

    function setProtectionSettings(bool _antiSnipe, bool
_antiBlock) external onlyOwner {
        initializer.setProtections(_antiSnipe, _antiBlock);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	MemeCasino.sol#L33,114,115,116,117,118,137,143,152,164,176
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
uint256 constant private startingSupply = 1_000_000
string constant private _name = "Meme Casino"
string constant private _symbol = "MEMEC"
uint8 constant private _decimals = 18
uint256 constant private _tTotal = startingSupply *
10**_decimals

Fees public _taxRates = Fees({
        buyFee: 500,
        sellFee: 500,
        transferFee: 0
    })


...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

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

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	MemeCasino.sol#L376,386
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor
piSwapPercent = priceImpactSwapPercent
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	MemeCasino.sol#L658,659,660
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 feeAmount = (tAmount * currentFee) / masterTaxDivisor
values.tFee = (feeAmount * ratios.reflection) / total
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	MemeCasino.sol#L324,517,636
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
address router
address constructorLP
uint256 initThreshold
uint256 initSwapAmount
ExtraValues memory values
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	MemeCasino.sol#L536
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
TOKEN.transfer(_owner, TOKEN.balanceOf(address(this)))
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-

	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFee OnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFee OnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupporting FeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
Initializer	Interface			
	setLaunch	External	✓	-
	getConfig	External	✓	-
	getInits	External	✓	-
	setLpPair	External	✓	-

MemeCasino	Implementation	IERC20		
		Public	Payable	-
	transferOwner	External	✓	onlyOwner
	renounceOwnership	External	✓	onlyOwner
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	approveContractContingency	External	✓	onlyOwner
	transferFrom	External	✓	-
	setNewRouter	External	✓	onlyOwner
	setLpPair	External	✓	onlyOwner
	setInitializer	Public	✓	onlyOwner
	isExcludedFromLimits	External		-
	setExcludedFromLimits	External	✓	onlyOwner

	isExcludedFromFees	External		-
	setExcludedFromFees	Public	✓	onlyOwner
	isExcludedFromProtection	External		-
	setExcludedFromProtection	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	lockTaxes	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	getTokenAmountAtPriceImpact	External		-
	setSwapSettings	External	✓	onlyOwner
	setPriceImpactSwapAmount	External	✓	onlyOwner
	setContractSwapEnabled	External	✓	onlyOwner
	excludePresaleAddresses	External	✓	onlyOwner
	_hasLimits	Internal		
	_transfer	Internal	✓	
	contractSwap	Internal	✓	inSwapFlag
	_checkLiquidityAdd	Internal	✓	
	sweepContingency	External	✓	onlyOwner
	sweepExternalTokens	External	✓	onlyOwner
	multiSendTokens	External	✓	onlyOwner
	isExcludedFromReward	Public		-
	setExcludedFromReward	Public	✓	onlyOwner
	tokenFromReflection	Public		-
	finalizeTransfer	Internal	✓	

| | takeTaxes | Internal | ✓ | |
| | _getRate | Internal | | |

Inheritance Graph
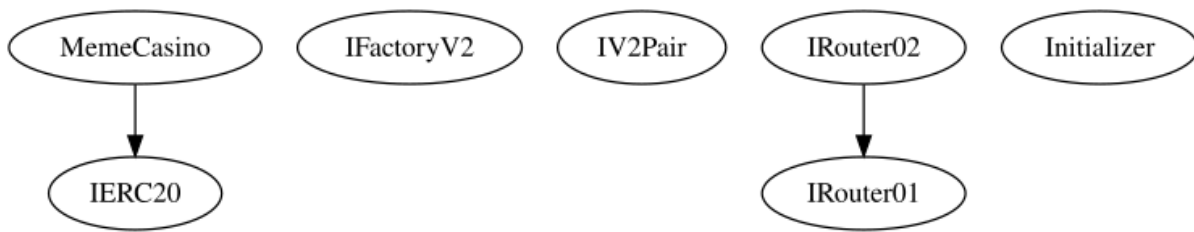
Flow Graph
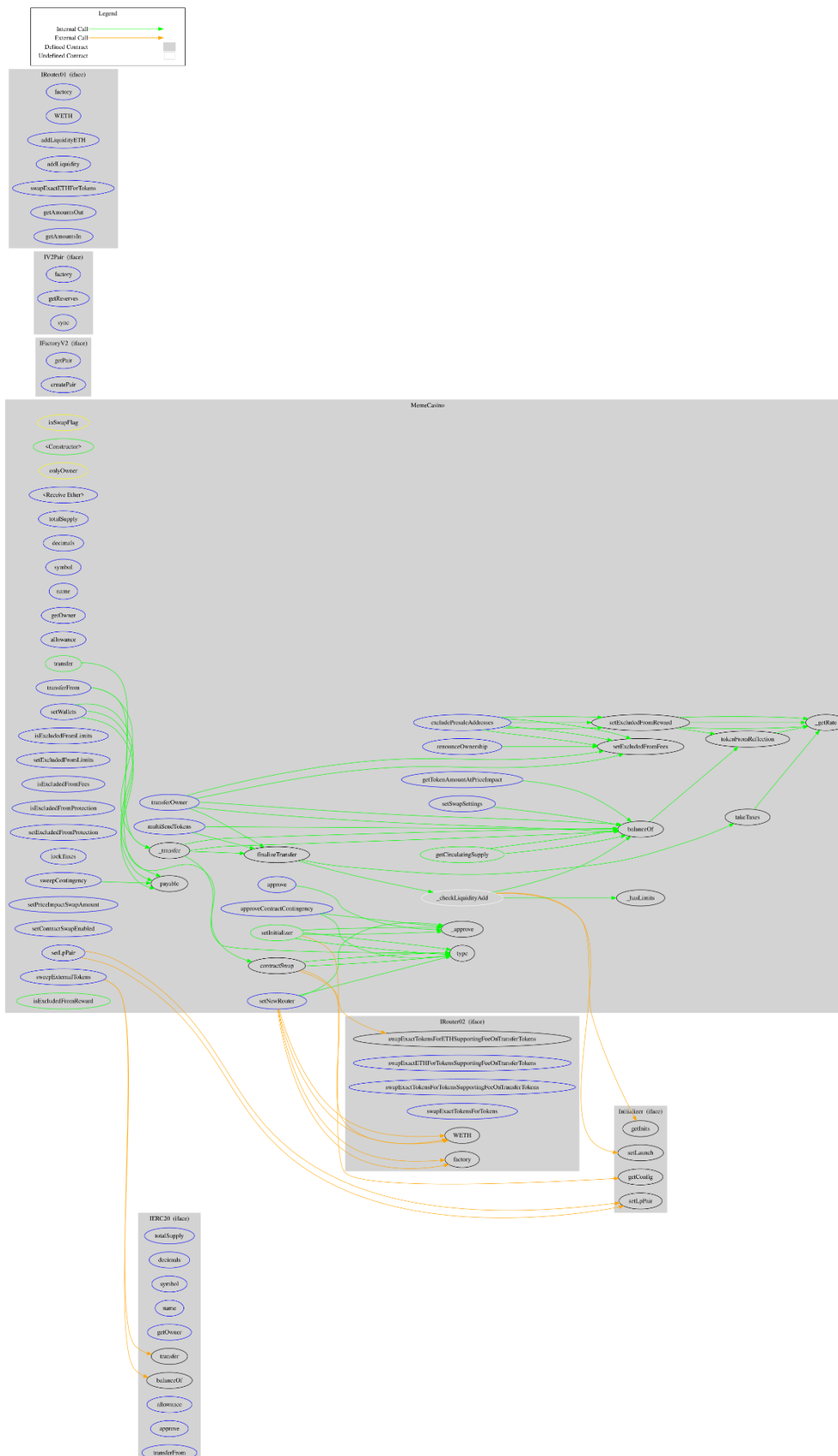
Summary

Meme Casino contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Meme Casino is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 5% fees on buy and sell transactions.

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