



Cyberscope

# Audit Report

## **Katzenomics 2.0**

September 2023

Network    ETH

Address    0xF6a67CC71c39400D4FB7DF4134b702115EB1aa77

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
●	RSML	Redundant SafeMath Library	Unresolved
●	RSK	Redundant Storage Keyword	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

# Table of Contents

<b>Analysis</b>	<b>1</b>
<b>Diagnostics</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Review</b>	<b>4</b>
Audit Updates	4
Source Files	4
<b>Findings Breakdown</b>	<b>5</b>
L02 - State Variables could be Declared Constant	6
Description	6
Recommendation	6
L04 - Conformance to Solidity Naming Conventions	7
Description	7
Recommendation	8
L07 - Missing Events Arithmetic	9
Description	9
Recommendation	9
L14 - Uninitialized Variables in Local Scope	10
Description	10
Recommendation	10
L20 - Succeeded Transfer Check	11
Description	11
Recommendation	11
<b>Functions Analysis</b>	<b>12</b>
<b>Inheritance Graph</b>	<b>17</b>
<b>Flow Graph</b>	<b>18</b>
<b>Summary</b>	<b>19</b>
<b>Disclaimer</b>	<b>20</b>
<b>About Cyberscope</b>	<b>21</b>

## Review

Contract Name	Katzenomics2
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	500 runs
Explorer	<a href="https://etherscan.io/address/0xf6a67cc71c39400d4fb7df4134b702115eb1aa77">https://etherscan.io/address/0xf6a67cc71c39400d4fb7df4134b702115eb1aa77</a>
Address	0xf6a67cc71c39400d4fb7df4134b702115eb1aa77
Network	ETH
Symbol	KATZEN2.0
Decimals	18
Total Supply	20,000,000,000

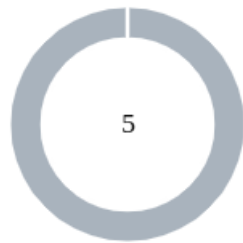
## Audit Updates

Initial Audit	27 Aug 2023 <a href="https://github.com/cyberscope-io/audits/blob/main/katzen20/v1/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/katzen20/v1/audit.pdf</a>
Corrected Phase 2	18 Sep 2023

## Source Files

Filename	SHA256
Staking.sol	493c3827b9afd0b33ca6ce0bbaa854ae427b3301d48f293b57dab0ad708cf070

## Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	5	0	0	0

## L02 - State Variables could be Declared Constant

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Katzenomics2.sol#L145
<b>Status</b>	Unresolved

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address payable public marketingWallet =  
payable(0x2D6089d74B52Af4045914758367c02205F1DBFfE)
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Katzenomics2.sol#L33,112,113,114,115,116,130,139,154,328
<b>Status</b>	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 = 20_000_000_000
string constant private _name = "Katzenomics 2.0"
string constant private _symbol = "KATZEN2.0"
uint8 constant private _decimals = 18
uint256 constant private _tTotal = startingSupply *
10**_decimals

Fees public _taxRates = Fees({
    buyFee: 0,
    sellFee: 200,
    transferFee: 0
})
uint256 constant masterTaxDivisor = 10000
bool public _hasLiqBeenAdded = false
bool _antiBlock

...
```

## 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Katzenomics2.sol#L353,363
<b>Status</b>	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.

## L14 - Uninitialized Variables in Local Scope

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Katzenomics2.sol#L297,492,523,524
<b>Status</b>	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 constructorLP
address router
uint256 initSwapAmount
uint256 initThreshold
bool checked
bool check
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Katzenomics2.sol#L511
<b>Status</b>	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
<b>IERC20</b>	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>IFactoryV2</b>	Interface			
	getPair	External		-
	createPair	External	✓	-
<b>IV2Pair</b>	Interface			
	factory	External		-

	getReserves	External		-
	sync	External	✓	-
<b>IRouter01</b>	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
<b>IRouter02</b>	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
<b>Initializer</b>	Interface			
	setLaunch	External	✓	-
	getConfig	External	✓	-
	getInits	External	✓	-
	setLpPair	External	✓	-

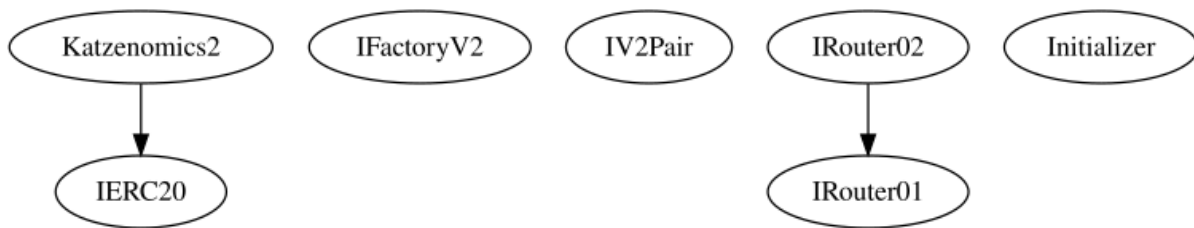
	checkUser	External	✓	-
	setProtections	External	✓	-
	removeSniper	External	✓	-
<b>Katzonomics2</b>	Implementation	IERC20		
		Public	Payable	-
	transferOwner	External	✓	onlyOwner
	renounceOwnership	External	✓	onlyOwner
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	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
	isExcludedFromFees	External		-
	setExcludedFromFees	Public	✓	onlyOwner
	isExcludedFromProtection	External		-
	setExcludedFromProtection	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	removeSniper	External	✓	onlyOwner
	setProtectionSettings	External	✓	onlyOwner
	lockTaxes	External	✓	onlyOwner
	setTaxes	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	✓	
	enableTrading	Public	✓	onlyOwner
	sweepContingency	External	✓	onlyOwner
	sweepExternalTokens	External	✓	onlyOwner
	multiSendTokens	External	✓	onlyOwner

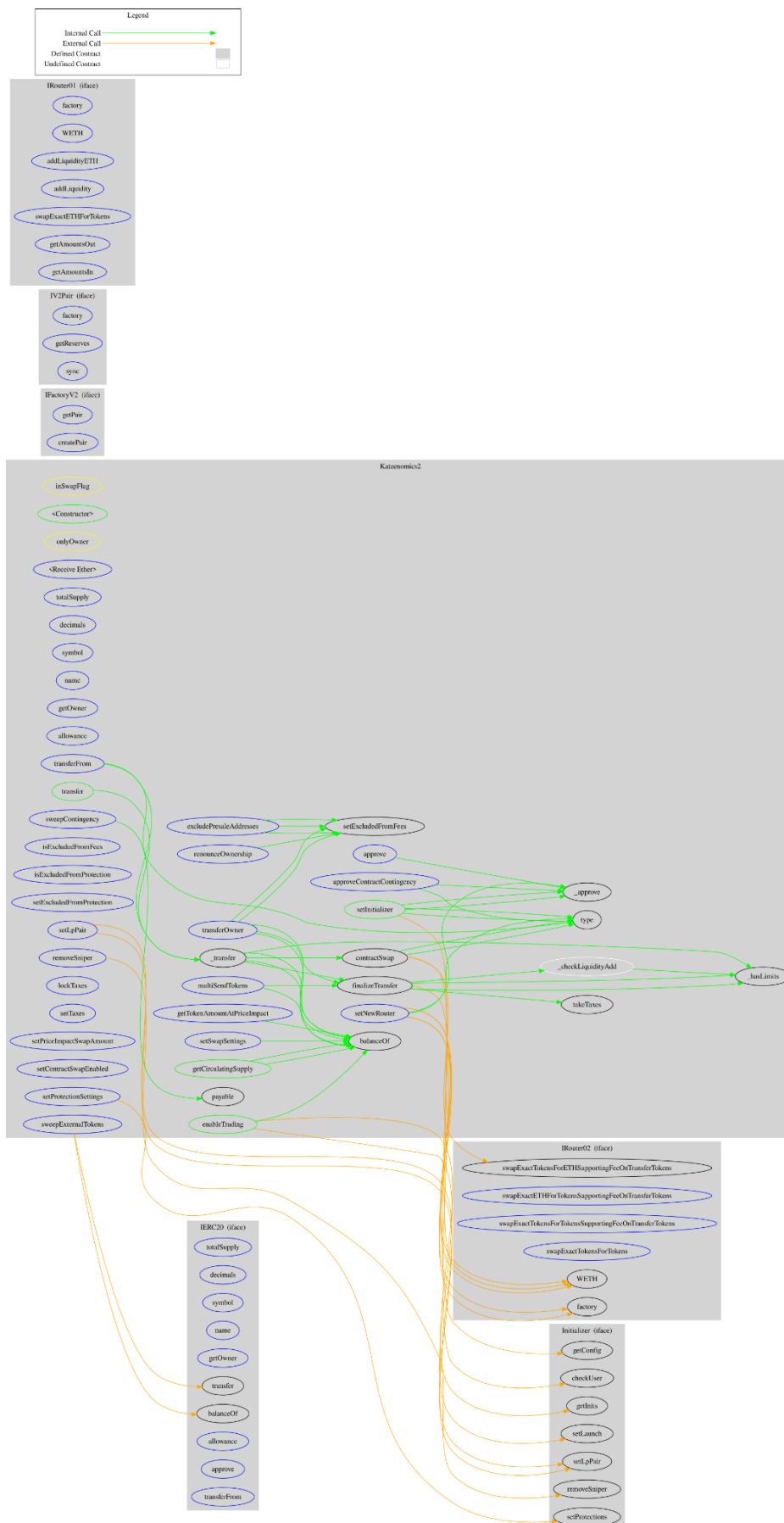


	finalizeTransfer	Internal	✓	
	takeTaxes	Internal	✓	

## Inheritance Graph



## Flow Graph



## Summary

Katzenomics 2.0 contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Katzenomics 2.0 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 10% fees.

## 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>