Cyberscope

Audit Report

Kazama Senshi

September 2022

Type	BEP20
Network	BSC
Address	0xa946CEB38E931554A570FA19D576493883FFE53E
Audited by	© cyberscope

Table of Contents

Table of Contents 1

Contract Review 3

Source Files 3

Audit Updates 3

Contract Analysis 4

ST - Stops Transactions 5

Description 5

Recommendation 5

ELFM - Exceeds Fees Limit 6

Description 6

Recommendation 6

MT - Mints Tokens 7

Description 7

Recommendation 7

Contract Diagnostics 8

ZD - Zero Division 9

Description 9

Recommendation 9

CO - Code Optimization 10

Description 10

Recommendation 10

L01 - Public Function could be Declared External 11

Description 11

Recommendation 11

L02 - State Variables could be Declared Constant 12

Description 12

Recommendation 12

L04 - Conformance to Solidity Naming Conventions 13

Description 13

Recommendation 13

L07 - Missing Events Arithmetic 14

Description 14

Recommendation 14

L09 - Dead Code Elimination 15

Description 15

Recommendation 15

L11 - Unnecessary Boolean equality 16

Description 16

Recommendation 16

L15 - Local Scope Variable Shadowing 17

Description 17

Recommendation 17

Contract Functions 18

Contract Flow 24

Domain Info 25

Summary 26

Disclaimer 27

About Cyberscope 28

Contract Review

Contract Name	KazamaSenshi
Compiler Version	v0.8.16+commit.07a7930e
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0xa946CEB38E931554A570FA19D576493883FFE53E
Symbol	KAZAMA
Decimals	18
Total Supply	285,500,001
Domain	https://kazamaswap.finance

Source Files

Filename	**SHA256**
contract.sol	30d247675e289d2bdeb69a68264569a9c856e87230d19f570bcc901266ee950a

Audit Updates

Initial Audit	17th September 2022
Corrected	

Contract 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	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Passed

ST - Stops Transactions

Criticality	medium
Location	contract.sol#L1444,1647
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the BurnPercentSettings to a high value and excluding the owner's wallet from the burn fees. As a result, the transactions will underflow since the burn amount will be greater than the transferred amount.

```
uint256 toReceive = amount - tokensToBurn;
```

The owner may also take advantage of it by setting the TotalFee to zero when the contract is applicable to swap. As a result, the transactions will stop since a zero division error will produced.

```
uint256 amountToLiquify = swapThreshold.mul(dynamicLiquidityFee).div(TotalFee).div(2);
```

Recommendation

Read more on the fees manipulation and zero division sections.

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceeds Fees Limit

Criticality	critical
Location	contract.sol#L1232
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the BurnPercentSettings function with a high value.

```
function burnPercentage(uint256 value) public view returns (uint256)  {
    uint256 roundValue = value.ceil(BurnPercentSettings);
    uint256 percentValue = roundValue.mul(BurnPercentSettings).div(100000); // =
2.75% (See line 911 for info).
    return perc
```

Recommendation

The contract could embody a check for the maximum acceptable value.

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

MT - Mints Tokens

Criticality	critical
Location	contract.sol#L1877
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the mint function. As a result the contract tokens will be highly inflated.

```
/// @notice Creates `_amount` token to `_to`. Must only be called by the owner
(SenshiMaster).
function mint(address _to, uint256 _amount) public OnlySenshiMaster {
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
}

/// @notice Creates `_amount` token to `_to`. Created especially for the bridge
contract.
function bridgeMint(address _to, uint256 _amount) public OnlyBridgeContract {
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	CO	Code Optimization	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

ZD - Zero Division

Criticality	medium
Location	contract.sol#L1647
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

```
uint256 amountToLiquify = swapThreshold.mul(dynamicLiquidityFee).div(TotalFee).div(2);
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

CO - Code Optimization

Criticality	minor / informative
Location	contract.sol#L1327
Status	Unresolved

Description

The contract contains statements that are not producing any effect in the logic.

```
shouldBurnSender(sender) ? true : false;
```

Recommendation

The empty statements could be eliminated.

L01 - Public Function could be Declared External

Criticality	minor / informative
Location	contract.sol#L397,408,416,423,433,441,448,460,468,1172,1179,1186,1246,1376,1395,1877,1883,1889,1905
Status	Unresolved

Description

Public functions that are never called by the contract should be declared external to save gas.

```
raiseSenshiMaster
raiseJin
recruitZaibatsu
setBridgeContract
removeSenshiMaster
removeZaibatsu
removeBridgeContract
renounceOwnership
transferOwnership
...
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contract.sol#L1057,1055,1056,873,885
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
BURN
BUSD
DEAD
WBNB
dividendsPerShareAccuracyFactor
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L284,285,286,287,312,323,332,341,350,807,909,864,872,873,874,1717,1726,1769,1812,1822,1827,1831,1843,1848,1054,1055,1056,1057,1058,1059,1060,1061,1065,1069,1070,1072,1076,1077,1078,1079,1080,1081,1082,1084,1085,1087,1088,1089,1090,1093,1094,1095,1096,1097,1099,1100,1101,1104,1107,1119,1877,1883,1919
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.

```
SenshiMaster
Jin
Zaibatsu
BridgeContract
OnlyOwner
OnlySenshiMaster
OnlyJin
OnlyZaibatsu
OnlyBridgeContract
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contract.sol#L909
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.

```
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	contract.sol#L558,568,587,601,647,657,620,630,505,533,674,245,263,230,237
Status	Unresolved

Description

Functions that are not used in the contract, and make the code's size bigger.

```
functionCall
functionCallWithValue
functionDelegateCall
functionStaticCall
isContract
sendValue
verifyCallResult
average
ceilDiv
...
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contract.sol#L1307,1419,1224
Status	Unresolved

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
shouldBurnSender(sender) == false
require(bool,string)(BuyBacker[_msgSender()] == true,)
```

Recommendation

Remove the equality to the boolean constant.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contract.sol#L1135
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
name
symbol
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
	max	Internal		
	min	Internal		
	average	Internal		
	ceil	Internal		
	ceilDiv	Internal		
TheZaibatsu	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	isSenshiMaster	Public		-
	isJin	Public		-

	isZaibatsu	Public		-
	isBridgeContract	Public		-
	raiseSenshiMaster	Public	✓	OnlyOwner
	raiseJin	Public	✓	OnlySenshiMaster
	recruitZaibatsu	Public	✓	OnlyJin
	setBridgeContract	Public	✓	OnlyJin
	removeSenshiMaster	Public	✓	OnlyJin
	removeZaibatsu	Public	✓	OnlyJin
	removeBridgeContract	Public	✓	OnlyJin
	renounceOwnership	Public	✓	OnlyJin
	transferOwnership	Public	✓	OnlyJin
	_transferOwnership	Internal	✓	
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-

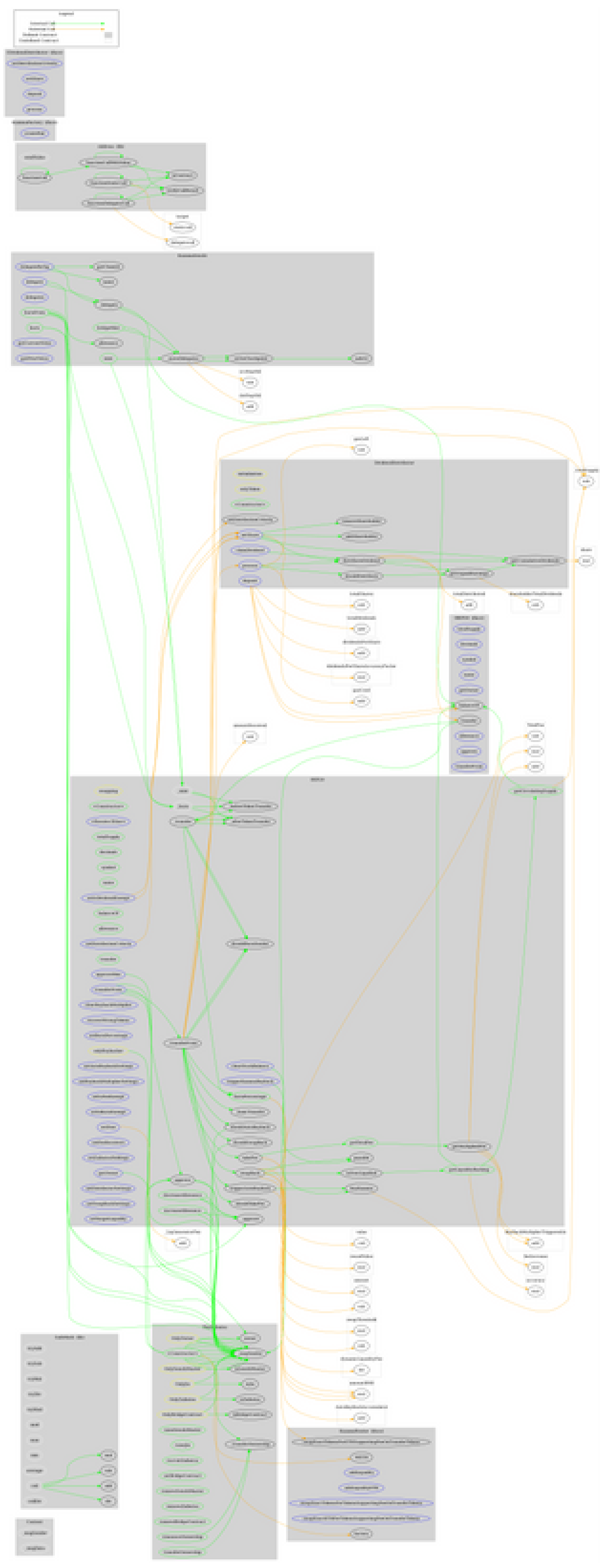
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
KazamaFactory	Interface			
	createPair	External	✓	-
KazamaRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	

	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
BEP20	Implementation	IBEP20, TheZaibatsu		
	<Constructor>	Public	Payable	-
	<Receive Ether>	External	Payable	-
	totalSupply	Public		-
	decimals	Public		-
	symbol	Public		-
	name	Public		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	Public		-
	burnPercentage	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	approveMax	External	✓	-
	transferFrom	External	✓	-
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
	shouldTakeFee	Internal		
	shouldBurnSender	Internal		

	getTotalFee	Public		-
	getMultipliedFee	Public		-
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	swapBack	Internal	✓	swapping
	shouldAutoBuyback	Internal		
	triggerKazamaBuyback	External	✓	OnlyZaibatsu
	clearBuybackMultiplier	External	✓	OnlyZaibatsu
	clearStuckBalance	External	✓	OnlyZaibatsu
	recoverWrongTokens	External	✓	OnlyJin
	setBurnPercentage	External	✓	OnlyZaibatsu
	setIsDividendExempt	External	✓	OnlyJin
	triggerAutoBuyback	Internal	✓	
	buyKazama	Internal	✓	swapping
	setAutoBuybackSettings	External	✓	OnlyZaibatsu
	setBuybackMultiplierSettings	External	✓	OnlyZaibatsu
	setIsFeeExempt	External	✓	OnlyZaibatsu
	setIsBurnExempt	External	✓	OnlyZaibatsu
	setFees	External	✓	OnlyJin
	setFeeReceivers	External	✓	OnlyJin
	setZaibatsuHoldings	External	✓	OnlyJin
	setDistributionCriteria	External	✓	OnlyZaibatsu
	setDistributorSettings	External	✓	OnlyZaibatsu
	setSwapBackSettings	External	✓	OnlyZaibatsu
	setTargetLiquidity	External	✓	OnlyJin
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
KazamaSenshi	Implementation	BEP20		
	mint	Public	✓	OnlySenshiMaster
	bridgeMint	Public	✓	OnlyBridgeContract
	burn	Public	✓	-

	burnFrom	Public	✓	-
	delegates	External		-
	delegate	External	✓	-
	delegateBySig	External	✓	-
	getCurrentVotes	External		-
	getPriorVotes	External		-
	_delegate	Internal	✓	
	_moveDelegates	Internal	✓	
	_writeCheckpoint	Internal	✓	
	safe32	Internal		
	getChainId	Internal		

Contract Flow



Domain Info

Domain Name	kazamaswap.finance
Registry Domain ID	8d4510f5c86046b7964acf67bec09b38-DONUTS
Creation Date	2022-03-10T12:30:39Z
Updated Date	2022-03-15T12:31:09Z
Registry Expiry Date	2023-03-10T12:30:39Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain was created 6 months before the creation of the audit. It will expire in 6 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

There are some functions that can be abused by the owner like stopping transactions, manipulating fees, minting tokens and burning tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burn functionality, then the users could lose their tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 7.5% fees.

Disclaimer

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io