



# Smart Contract Security Audit

**Project: Kazama Swap**

Sep 14, 2022



**Contract Address**

0xa946CEB38E931554A570FA19D576493883FFE53E

# Table of Contents

- 1 Disclaimer**
- 2 Audit Review**
- 3 Project Review**
- 4 Smart Contract Vulnerability Checks**
- 5 Manual Code Review**
- 6 Owner Privileges**
  - 6.1 Contract Ownership
  - 6.2 Liquidity Overview
- 7 Tokenomics**
- 8 Social Media Check**
- 9 Website Review**
- 10 Audit Conclusion**

## Disclaimer

The contents of this report reflect only the CRACKEN TECH audit team's understanding of the current progress and status of the security of the code audited, to verify the integrity of the code provided for the scope of this audit. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit are recommended after the issues covered are fixed. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

The review does not address the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from off-chain sources are not extended by this review either.

## Audit Review

The source code of the Kazama Swap was audited in order to acquire a clear impression of how the project was implemented. The Cracken Tech audit team conducted in-depth research, analysis, and scrutiny, resulting in a series of observations. A detailed list of each issue found, and vulnerabilities in the source code will be included in the audit report. The problems and potential solutions are given in this report, we will identify common sources for such problems and comments for improvement.

The auditing process will follow a routine as special considerations by Cracken:

- Review of the specifications, sources, and instructions provided to Cracken to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Cracken describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
- Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

## Project Review

### Token Summary

Parameter	Result
Token Name	Kazama Senshi
Token Symbol	KAZAMA
Token Decimal	18
Total Supply	285,500,001
Platform	BSC
Buy Tax Fee	13%
Sell Tax Fee	13%
Contract Creation Date	Sep 09, 2022
Liquidity Status	Not available when auditing
Liquidity Lockup Time	Not available when auditing
Compiler Version	v0.8.16+commit.07a7930e
Optimization	Yes with 200 runs
Contract Address	0xa946CEB38E931554A570FA19D576493883FFE53E
Deployer Address	0xe2401fa0402b7ade232c149dd2db2edde90f2dc4
Owner Address	0xe2401fa0402b7ade232c149dd2db2edde90f2dc4

### Source Code

CRACKEN was commissioned by Kazama Swap to perform an audit based on the following smart contract:

<https://bscscan.com/address/0xa946CEB38E931554A570FA19D576493883FFE53E>





## Smart Contract Vulnerability Checks

Vulnerability	Auto-Scan	Manual-Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	Low / No Risk
Code With No Effects	Complete	Complete	Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	Low / No Risk
Hash Collisions with Multiple Variable Length Arguments	Complete	Complete	Low / No Risk
Unexpected Ether balance	Complete	Complete	Low / No Risk
Presence of unused variables	Complete	Complete	Low / No Risk
Right-To-Left-Override control character (U+202E)	Complete	Complete	Low / No Risk
Typographical Error	Complete	Complete	Low / No Risk
DoS With Block Gas Limit	Complete	Complete	Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	Low / No Risk
Insufficient Gas Grieving	Complete	Complete	Low / No Risk
Incorrect Inheritance Order	Complete	Complete	Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	Low / No Risk
Requirement Violation	Complete	Complete	Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	Low / No Risk
Weak Sources of Randomness from Chain Attributes	Complete	Complete	Low / No Risk
Authorization through tx. origin	Complete	Complete	Low / No Risk
Delegate call to Untrusted Callee	Complete	Complete	Low / No Risk





Vulnerability	Auto-Scan	Manual-Scan	Result
Use of Deprecated Solidity Functions	Complete	Complete	Low / No Risk
Assert Violation	Complete	Complete	Low / No Risk
Reentrancy	Complete	Complete	Low / No Risk
Unprotected SELF-DESTRUCT Instruction	Complete	Complete	Low / No Risk
Unprotected Ether Withdrawal	Complete	Complete	Low / No Risk
Outdated Compiler Version	Complete	Complete	Low / No Risk
Integer Overflow and Underflow	Complete	Complete	Low / No Risk
Function Default Visibility	Complete	Complete	Low / No Risk

## Manual Code Review

### Classification of Issues

Severity	Description
 High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
 Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
 Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
 Informational	A vulnerability that has an informational character but is not affecting any of the code.

### Findings

Severity	Found
 High-Risk	1
 Medium-Risk	0
 Low-Risk	0
 Informational	1
Total	2



● High-Risk: functions make cause the rug or scam project. **Must be fixed.**

## Set mint function is enabled

Description:

**The owner can mint the new tokens after the initial deployed.**

**[HIGH-RISK]**

```
function _mint(address account, uint256 amount) internal virtual {  
  
    require(account != address(0), "ERC20: mint to the zero address");  
  
    _beforeTokenTransfer(address(0), account, amount);  
  
    _totalSupply += amount;  
    _balances[account] += amount;  
    emit Transfer(address(0), account, amount);  
  
    _afterTokenTransfer(address(0), account, amount);  
}  
  
/// @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);  
}
```

### **Comments:**

**The owner told us that the new tokens will be minted to be paid out to stakers, yield farmers, etc. They will renounce the ownership of their platform on the main net.**

● Informational: Implementation of certain corrective actions or accepting the risk.

## Set max buy / sell tax fee

Description:

**The owner can change the buy & sell fees up to 7.69%**

```
function setFees(uint256 _LiqGeneratorFee, uint256 _BuyBackBurnFee, uint256
_TreasuryFee, uint256 _RewardsFee, uint256 _FeeDenominator) external OnlyJin {

    LiqGeneratorFee = _LiqGeneratorFee;

    BuyBackBurnFee = _BuyBackBurnFee;

    TreasuryFee = _TreasuryFee;

    RewardsFee = _RewardsFee;

    TotalFee =

_LiqGeneratorFee.add(_BuyBackBurnFee).add(_TreasuryFee).add(_RewardsFee);

    FeeDenominator = _FeeDenominator;

    require(TotalFee < FeeDenominator/13);

}
```

## Privileged Functions

**onlyOwner**

Function Name	Parameters	Visibility
approve	address spender, uint256 amount	External
approveMax	address spender	External
bridgeMint	address _to, uint256 _amount	Public
burn	uint256 amount	Public
burnFrom	address account, uint256 amount	Public
clearBuybackMultiplier	None	External
clearStuckBalance	uint256 amountPercentage	External
decreaseAllowance	address spender, uint256 subtractedValue	Public
delegate	address delegatee	Public
delegateBySig	address delegatee, uint nonce, uint expiry, uint8 v, bytes32 r, bytes32 s	External
increaseAllowance	address spender, uint256 addedValue	Public
mint	address account, uint256 amount	Internal
raiseJin	address adr	Public
raiseSenshiMaster	address adr	Public
recoverWrongTokens	address _tokenAddress, uint256 _tokenAmount	External
recruitZaibatsu	address adr	Public
removeBridgeContract	address adr	Public

removeSenshiMaster	address adr	Public
removeZaibatsu	address adr	Public
renounceOwnership	None	Public
setAutoBuybackSettings	bool _enabled, uint256 _cap, uint256 _amount, uint256 _period	External
setBridgeContract	address adr	Public
setBurnPercentage	uint256 _BurnPercentSettings	External
setBuybackMultiplierSettings	int256 numerator, uint256 denominator, uint256 length	External
setDistributionCriteria	uint256 _minPeriod, uint256 _minDistribution	External
setDistributorSettings	uint256 gas	External
setFeeReceivers	address _LiquidityReceiver, address _TreasuryReceiver	External
setFees	uint256 _LiqGeneratorFee, uint256 _BuyBackBurnFee, uint256 _TreasuryFee, uint256 _RewardsFee, uint256 _FeeDenominator	External
setIsBurnExempt	address holder, bool exempt	External
setIsDividendExempt	address holder, bool exempt	External
setIsFeeExempt	address holder, bool exempt	External
setSwapBackSettings	bool _enabled, uint256 _amount	External
setTargetLiquidity	uint256 _target, uint256 _denominator	External
setZaibatsuHoldings	address _ZaibatsuHoldings	External
transfer	address recipient, uint256 amount	External
transferFrom	address sender, address recipient, uint256 amount	Public
transferOwnership	address newOwner	Public
triggerKazamaBuyback	uint256 amount, bool triggerBuybackMultiplier	External

## Contract Ownership

The contract ownership of Kazama Swap is not currently being renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address `0xe2401fa0402b7ade232c149dd2db2edde90f2dc4` which can be viewed: [HERE](#)

The owner wallet has the power to call the functions displayed on the privileged functions list above, if the owner wallet is compromised these privileges could be exploited.

We recommend the team renounce ownership at the right timing if possible, or gradually migrate to a time lock with governing functionalities in respect of transparency and safety considerations.

## Liquidity Overview

### Liquidity Information

Parameter	Result
Pair Address	<code>0xef731d7cc3a738b8ba57f6f9aea5cc17df402e92</code>
KAZAMA Reserves	0.00 KAZAMA
BNB Reserves	0.00 BNB
Liquidity Value	\$0.00 USDT
Liquidity Ownership	The token does not have liquidity at the moment of the audit

## Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0xe2401fa0402b7ade232c149dd2db2edde90f2dc4	285,500,001	100.0000%

## Social Media Check

Social Media Type	Link	Result
Website	<a href="https://kazamaswap.finance/">https://kazamaswap.finance/</a>	Checked
Twitter	<a href="https://twitter.com/KazamaSwap/">https://twitter.com/KazamaSwap/</a>	Checked
Telegram	<a href="https://t.me/KazamaSwap/">https://t.me/KazamaSwap/</a>	Checked
Discord	<a href="https://discord.com/invite/s49n7XBkXz/">https://discord.com/invite/s49n7XBkXz/</a>	Checked

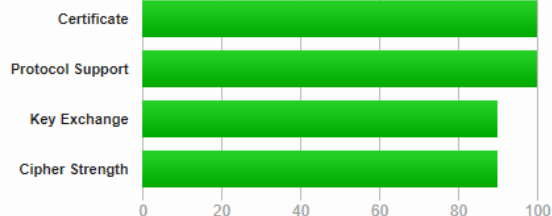
## Website Review



The screenshot shows the KAZAMASWAP website interface. The top navigation bar includes a menu icon, the KAZAMASWAP logo, a balance of \$0.00001, a settings gear, and a BNB Testnet dropdown. The left sidebar lists various features: Exchange (Swap Tokens, Liquidity), Earn (Staking Pools, Yield Farms), Win (Kazama Lottery), Predictions, NFT Market (Marketplace, Collections), Launchpad (Create Presale, Create IFO), and Governance (All Proposals, Create Proposal). The main content area is titled "STAKING POOLS" with the subtitle "STAKE TO EARN TOKENS". It features a large "APPLY FOR A POOL" button and filters for User Status (All Pools), Pool Status (Live Pools), and Sort By (Hot). Below these are two pool cards. The first card is for "Stake KAZAMA" with a recent profit of 208,725.68, a locked amount of 353,720.39, an APY of 2,982.70%, and a total staked amount of 3,516,589 KAZAMA. It shows a "Locked" status and a "Yield Boost" of 1.36x. The second card is for "Flexible KAZAMA" with a recent profit of 1,285,680.82, a staked amount of 3,901,474.91, an APY of 2,149.42%, and a total staked amount of 3,901,475 KAZAMA. It shows a "Staked" status. Both cards have "ADD KAZAMA" and "EXTEND" buttons.

### Summary

#### Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server supports TLS 1.3.



## Certificate #1: RSA 2048 bits (SHA256withRSA)



### Server Key and Certificate #1



Subject	kazamaswap.finance Fingerprint SHA256: cba1132686bc710fac5a98dd102d4f1ca4a9cc9242656b3728a9da581bf9248 Pin SHA256: g+bEHQ74fgwKDKbqobK8wGdV3m9r4qjWDjll//ugEAjE=
Common names	kazamaswap.finance
Alternative names	kazamaswap.finance
Serial Number	043474e8990f9f8b33db1c78dd97351136f8
Valid from	Fri, 05 Aug 2022 19:14:08 UTC
Valid until	Thu, 03 Nov 2022 19:14:07 UTC (expires in 1 month and 20 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	R3 AIA: <a href="http://r3.i.lencr.org/">http://r3.i.lencr.org/</a>
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation information	OCSP OCSP: <a href="http://r3.o.lencr.org">http://r3.o.lencr.org</a>
Revocation status	Good (not revoked)
DNS CAA	No ( <a href="#">more info</a> )
Trusted	Yes Mozilla Apple Android Java Windows

- Mobile friendly
- Contains no code errors
- SSL is secured
- No spelling errors

## Audit Conclusion

- The owner cannot pause trading.
  - **The owner cannot mint new tokens [High-Risk].**
  - The owner cannot add blacklist users.
  - The owner cannot set the max transaction amount.
  - The owner can change the buy/sell fee up to 7.69%.
- (All functions cannot be used if the ownership is renounced)

**AUDIT IS PASSED**