Smart Contract Audit Report

Audit was conducted on the **SPEEZY** Smart Contract

Smart Contract	**SPEEZY**
Type Of Utility	ERC20
Platform	BSC, Ethereum Virtual Machine
Chain Id	56
Language	Solidity 0.8.7
Address	0xbB560523D3c2F34C5db58C3119b4828C7d96710a

Audit Score

Section	Score
Codebase Security	**100%**
Codebase Complexity and Practices	**95%**
Owner Privileges and Control	**100%**
Overall Score	**98.3%**

Branding:

Website Telegram Twitter

Liquidity Burned Renounced

Content

Scope of the audit

This Audit Report mainly focuses on the overall security of the **AK47 INU** token Smart Contract. This audit was conducted with rigorous attention to the general implementation of the contract and by examining the overall architectural layout of the software implementation. The reliability and correctness of this smart contract's codebase are being assessed.

Security Scope

Identifies security related issues within each contract and the system of contract.

General Code Quality

A full assessment of the code quality and general software architecture patterns and best practices used.

Auditing Methods Used

Rigorous testing of the project has been performed. Detailed code base analysis was conducted, reviewing the smart contract architecture to ensure it is structured and safe.

A detailed, line by line inspection of the codebase was conducted to find any potential security vulnerabilities such as denial of service attacks, race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Automated and manual testing was employed that included:
– Analysis of on-chain data security
– Analysis of the code in-depth and detailed, manual review of the code, line-by-line.
– Deployment of the code on an in-house testnet blockchain and running live tests●
– Determining failure preparations and if worst-case scenario protocols are in place
– Analysis of any third-party code use and verifying the overall security of this

Tools Used:
Remix IDE, Ganache, SolHint, VScode, Mythril, Contract Library Hardhat

Assessing Possible Issues

Any issue detected during the conduction of this audit will be categorized under one of 3 severity levels: low, medium, and high.

Low level Severity Issues

Issues that do not pose any serious threat to the functionality of the software.

Medium level Severity issues

Issues that can cause potential problems to the overall health of the software application but that can be fixed without having any breaking changes on the current functionality.

High level Severity issues

Critical issues that affect the smart contract's overall performance and functionality. These issues should be fixed urgently.

General Issues Report

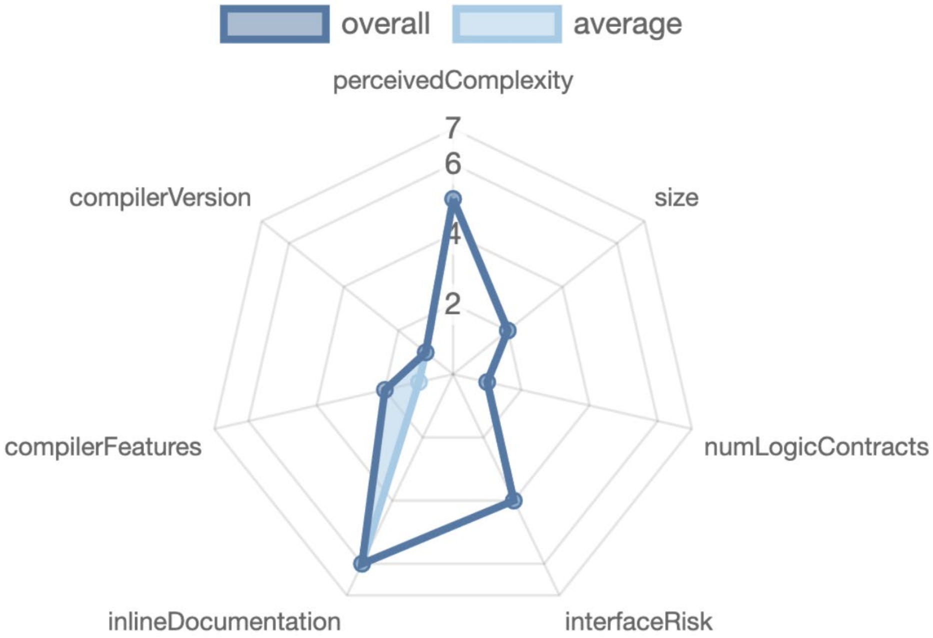
General issues that were found during manual and automatic assessments

No	Issue Verification	Status
1	Compiler warnings	Passed
2	Reentrancy and Race Conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	DoS with block gas limit.	Passed
7	DoS with Revert.	Passed
8	Timestamp dependence.	Passed
9	Methods execution permissions.	Passed
10	Economy model.	Passed
11	The impact of the exchange rate on the logic.	Passed
12	Private user data leaks.	Passed
13	Scoping and Declarations.	Passed
14	Arithmetic accuracy.	Passed

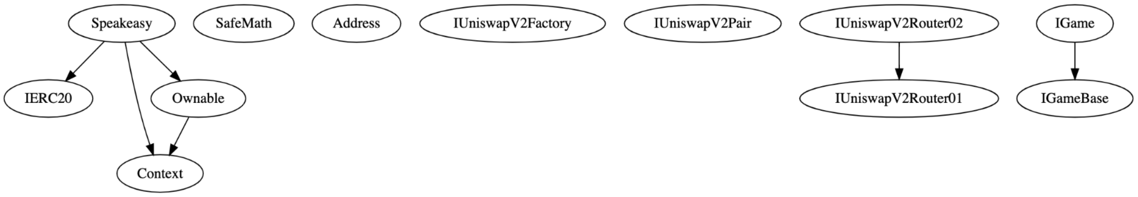
Issues Found

Low Level Severity	Medium Level Severity	High Level Severity
0	0	0

Risk Assessment



Contract Dependency Graphs



Manual Code Inspection

The code of the target contract and its dependencies was reviewed, deployed, and manually tested by our developers.

No	Contract	Issues
1	Speezy	2
2	Ownable	None
3	Context	None

Issues Found

Low Level Severity	Medium Level Severity	High Level Severity
1	2	0

Unprovided Code

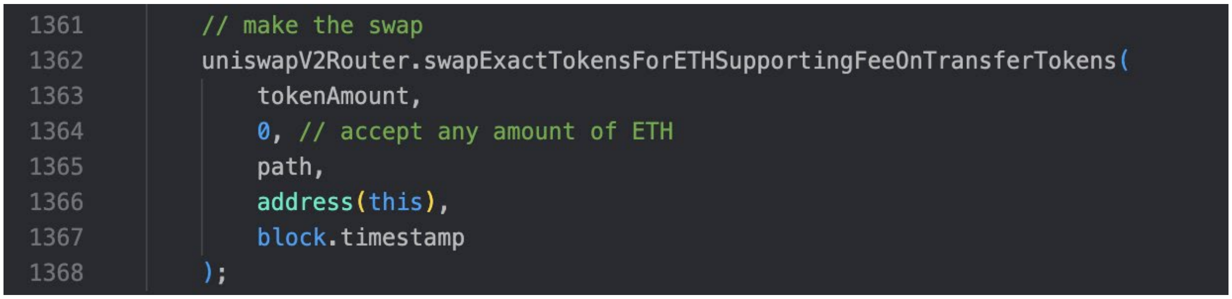
No	Contract
1	Game.sol

Inspections

Contract	Speezy
Address	0x5a54c3266634a4B7560cB435c36Df6652529ffD7
Issues	3
Notes	ERC-20 Token

Issues

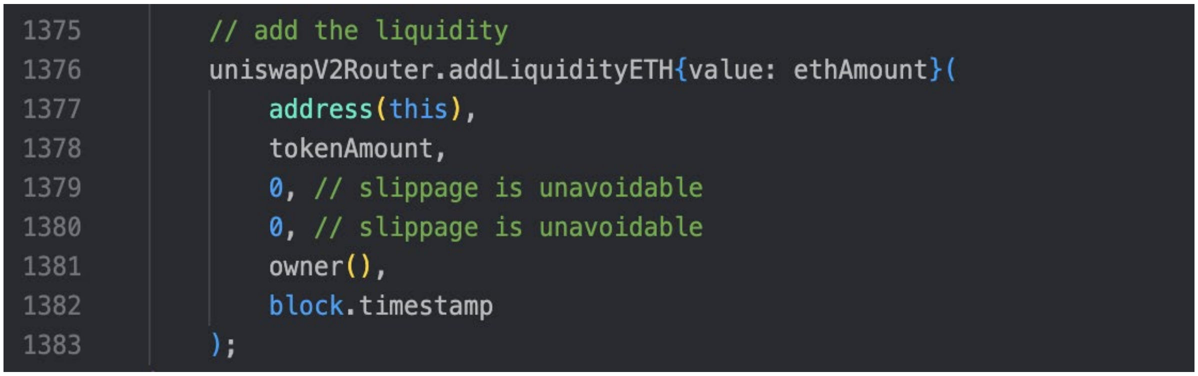
1. Front Running Attack Surface



```
1361        // make the swap
1362        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1363            tokenAmount,
1364            0, // accept any amount of ETH
1365            path,
1366            address(this),
1367            block.timestamp
1368        );
```

Line	**1362**
Severity	**Medium**
Method	**swapExactTokensForETHSupportingFeeOnTransferTokens (uint256 tokenAmount, address _to)**
Description	**Setting the minimum expect output amount for a swap to be 0 can lead to frontrunning attacks that especially if there are high volume transactions involved.**
Notes	**Calculate and set a minimum output amount or limit the max transaction amount to reduce the attack probability.**

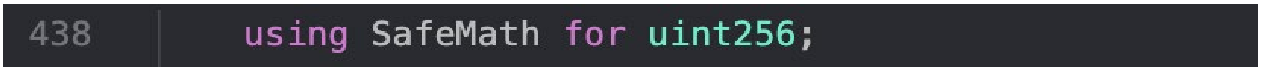
2. Liquidity Centralization



```
1375        // add the liquidity
1376        uniswapV2Router.addLiquidityETH{value: ethAmount}(
1377            address(this),
1378            tokenAmount,
1379            0, // slippage is unavoidable
1380            0, // slippage is unavoidable
1381            owner(),
1382            block.timestamp
1383        );
```

Line	**1376**
Severity	**Medium**
Method	**uniswapV2Router.addLiquidityETH**
Description	**All the newly minted liquidity is redirected to the owner's address**
Notes	**Minted liquidity should be stored in a non centralized environment, locked or burned**

3. Redundant use of library



```
438        using SafeMath for uint256;
```

Line	**438**
Severity	**Low**
Method	**—**
Description	**Version > 0.8.0 of the solidity compiler has built in math safety.**
Notes	**This library can be removed**

Access Control and Privileges

The contract uses a single owner access control system for setting contract specific parameters.

Speezy.sol

Role	Privileges

Owner	`renounceOwnership`
	`transferOwnership`
	`setExclusionFromRewards`
	`setExclusionFromFees`
	`setSwapTokensBNBLimit`
	`setMaxSellAllowanceBeforeExtra`
	`setMaxTxPercent`
	`removeBlacklist`
	`setSwapAndLiquifyEnabled`
	`setFees`
	`setAddressFee`
	`setTaxWallets`
	`enableTrading`
	`setJackpotGame`
	`removeJackpotGame`
	`setExclusionToAddrFromCredits`
	`setExclusionFromAddrFromCredits`
	`setExclusionToAddrFromCreditRemoval`
	`setCreditPrice`
	`setAutoSend`
	`withdrawBNB`
	`withdrawToken`
	`updateUniswapV2Router`
	`setOverSellAddress`
	`setTaxesToRedistribute`
	`setPresaleAddress`

Restrictions that the owner can impose:

- Exclude accounts from fess
- Set fees
- Set individual fees on addresses
- Halt trading
- Block addresses from receiving or sending transactions
- Limit trading frequency

Notes

The owner of this contract can censor/restrict parties from accessing this contract's functionality.

Conclusion

The **Speezy** Smart contracts do not contain any high severity issues!

Audit Score

Section	Score
Codebase Security	**100%**
Codebase Complexity and Practices	**93%**
Owner Privileges and Control	**95%**
Overall Score	**96%**

Speezy has passed the Smart Contract Audit by HedgePay Sdn Bhd

KYC Verifications: N/A

Smart contract Audit completion: 2nd^{t} Sept 2022. 12:00 pm UTC

https://github.com/HedgePay/audits

CERTIFIED

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. To get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us based on what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HedgePay and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HedgePay) owe no duty of care towards you or any other person, nor does HedgePay make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HedgePay hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HedgePay hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HedgePay, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.