KISEKI

# Smart Contract Audit Report

SYSFIXED

29 Jan 2023

# TABLE OF CONTENTS

# AUDITED DETAILS

## Audited Project

| Project name | Token ticker | Blockchain |
|---|---|---|
| KISEKI | KISEKI | Binance Smart Chain |

## Addresses

| | |
|---|---|
| **Contract address** | 0xD60F5491460903D090cE602E47b0BbF91eb5Da57 |
| **Contract deployer address** | 0x694981b6F83fea88C2Bbd1b7BAEEd9FA2330e5b4 |

## Project Website

**https://www.kisekiwallet.net/**

## Codebase

**https://bscscan.com/address/0xD60F5491460903D090cE602E47b0BbF91eb5Da57#code**

# SUMMARY

Revolutionize the way you access Web3 Kiseki Wallet is a powerful and beautifully designed, all-in-one DeFi tool, which will give traders an advantage in speed, security, and versatility, through a variety of unique features its ZERO tax, Snipe Tokens, Multi-Wallet Functionality & Swap, Future Features: In-Wallet Governance, Launchpad, Release on iOS & Android, TG, YouTube, TikTok, Exclusive NFT Airdrop, for the FIRST 250 qualified buyers, Demo Live.

## Contract Summary

**Documentation Quality**

KISEKI provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

**Code Quality**

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by KISEKI with the discovery of several low issues.

**Test Coverage**

Test coverage of the project is 100% ( Through Codebase )

## Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 174, 191, 207, 229, 231, 243, 244, 655, 655, 655, 656, 733, 774, 774, 776, 776, 780, 807, 807, 811, 811 and 822.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 791 and 792.

# CONCLUSION

We have audited the NamaProject Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in NamaProject Project. This process is used to find bugs, technical issues, and security loopholes that find some common issues in the code.

The security audit report provides a satisfactory result with some low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. Some of the low issues were Out of bounds array access. Be aware The index access expression can cause an exception in case of the use of an invalid array index value.

# AUDIT RESULT

| Article | Category | Description | Result |
|---|---|---|---|
| Default Visibility | SWC-100<br>SWC-108 | Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously. | PASS |
| Integer Overflow and Underflow | SWC-101 | If unchecked math is used, all math operations should be safe from overflows and underflows. | ISSUE FOUND |
| Outdated Compiler Version | SWC-102 | It is recommended to use a recent version of the Solidity compiler. | PASS |
| Floating Pragma | SWC-103 | Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. | PASS |
| Unchecked Call Return Value | SWC-104 | The return value of a message call should be checked. | PASS |
| Unprotected Ether Withdrawal | SWC-105 | Due to missing or insufficient access controls, malicious parties can withdraw from the contract. | PASS |
| SELFDESTRUCT Instruction | SWC-106 | The contract should not be self-destructible while it has funds belonging to users. | PASS |
| Reentrancy | SWC-107 | Check effect interaction pattern should be followed if the code performs recursive call. | PASS |
| Uninitialized Storage Pointer | SWC-109 | Uninitialized local storage variables can point to unexpected storage locations in the contract. | PASS |
| Assert Violation | SWC-110<br>SWC-123 | Properly functioning code should never reach a failing assert statement. | ISSUE FOUND |
| Deprecated Solidity Functions | SWC-111 | Deprecated built-in functions should never be used. | PASS |
| Delegate call to Untrusted Callee | SWC-112 | Delegatecalls should only be allowed to trusted addresses. | PASS |

| | | | |
|---|---|---|---|
| DoS (Denial of Service) | SWC-113<br>SWC-128 | Execution of the code should never be blocked by a specific contract state unless required. | PASS |
| Race Conditions | SWC-114 | Race Conditions and Transactions Order Dependency should not be possible. | PASS |
| Authorization through tx.origin | SWC-115 | tx.origin should not be used for authorization. | PASS |
| Block values as a proxy for time | SWC-116 | Block numbers should not be used for time calculations. | PASS |
| Signature Unique ID | SWC-117<br>SWC-121<br>SWC-122 | Signed messages should always have a unique id. A transaction hash should not be used as a unique id. | PASS |
| Incorrect Constructor Name | SWC-118 | Constructors are special functions that are called only once during the contract creation. | PASS |
| Shadowing State Variable | SWC-119 | State variables should not be shadowed. | PASS |
| Weak Sources of Randomness | SWC-120 | Random values should never be generated from Chain Attributes or be predictable. | PASS |
| Write to Arbitrary Storage Location | SWC-124 | The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations. | PASS |
| Incorrect Inheritance Order | SWC-125 | When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/. | PASS |
| Insufficient Gas Griefing | SWC-126 | Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract. | PASS |
| Arbitrary Jump Function | SWC-127 | As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value. | PASS |

# SMART CONTRACT ANALYSIS

| | |
|---|---|
| **Started** | Saturday Jan 28 2023 03:26:39 GMT+0000 (Coordinated Universal Time) |
| **Finished** | Sunday Jan 29 2023 09:53:22 GMT+0000 (Coordinated Universal Time) |
| **Mode** | Standard |
| **Main Source File** | KISEKI.sol |

## Detected Issues

| ID | Title | Severity | Status |
|---|---|---|---|
| SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED | low | acknowledged |

| | | | |
|---|---|---|---|
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED | low | acknowledged |
| SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED | low | acknowledged |
| SWC-110 | OUT OF BOUNDS ARRAY ACCESS | low | acknowledged |
| SWC-110 | OUT OF BOUNDS ARRAY ACCESS | low | acknowledged |

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 174

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
173   unchecked {
174   _approve(sender, _msgSender(), currentAllowance - amount);
175   }
176   }
177
178
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 191

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
190   spender,
191   _allowances[_msgSender()][spender] + addedValue
192   );
193   return true;
194   }
195
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 207

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
206   unchecked {
207   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
208   }
209
210   return true;
211
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 229

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
228   unchecked {
229   _balances[sender] = senderBalance - amount;
230   }
231   _balances[recipient] += amount;
232
233
```

# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 231

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
230   }
231   _balances[recipient] += amount;
232
233   emit Transfer(sender, recipient, amount);
234
235
```

# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 243

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
242
243   _totalSupply += amount;
244   _balances[account] += amount;
245   emit Transfer(address(0), account, amount);
246
247
```

# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 244

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
243   _totalSupply += amount;
244   _balances[account] += amount;
245   emit Transfer(address(0), account, amount);
246
247   _afterTokenTransfer(address(0), account, amount);
248
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 655

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
654
655   _mint(owner(), 10**8 * (10**18));
656   swapTokensAtAmount = totalSupply() / 5000;
657   }
658
659
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 655

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
654
655   _mint(owner(), 10**8 * (10**18));
656   swapTokensAtAmount = totalSupply() / 5000;
657   }
658
659
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 655

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
654
655   _mint(owner(), 10**8 * (10**18));
656   swapTokensAtAmount = totalSupply() / 5000;
657   }
658
659
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 656

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
655   _mint(owner(), 10**8 * (10**18));
656   swapTokensAtAmount = totalSupply() / 5000;
657   }
658
659   receive() external payable {}
660
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 733

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
732   require(
733   newAmount > totalSupply() / 100000,
734   "SwapTokensAtAmount must be greater than 0.001% of total supply"
735   );
736   swapTokensAtAmount = newAmount;
737
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 774

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
773   if (from == uniswapV2Pair) {
774   fees = (amount * feeOnBuy) / 100;
775   } else if (to == uniswapV2Pair) {
776   fees = (amount * feeOnSell) / 100;
777   } else {
778
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 774

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
773   if (from == uniswapV2Pair) {
774   fees = (amount * feeOnBuy) / 100;
775   } else if (to == uniswapV2Pair) {
776   fees = (amount * feeOnSell) / 100;
777   } else {
778
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 776

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
775   } else if (to == uniswapV2Pair) {
776   fees = (amount * feeOnSell) / 100;
777   } else {
778   fees = 0;
779   }
780
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 776

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
775   } else if (to == uniswapV2Pair) {
776   fees = (amount * feeOnSell) / 100;
777   } else {
778   fees = 0;
779   }
780
```

# SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 780

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
779   }
780   amount -= fees;
781   if (fees > 0) {
782   super._transfer(from, address(this), fees);
783   }
784
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 807

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
806   payable(marketingWallet),
807   (addressBalance * marketingShare) / SHAREDIVISOR
808   );
809   sendBNB(
810   payable(devWallet),
811
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 807

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
806   payable(marketingWallet),
807   (addressBalance * marketingShare) / SHAREDIVISOR
808   );
809   sendBNB(
810   payable(devWallet),
811
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 811

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
810   payable(devWallet),
811   (addressBalance * teamShare) / SHAREDIVISOR
812   );
813
814   emit SwapAndSendFee(tokenAmount, newBalance);
815
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 811

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
810   payable(devWallet),
811   (addressBalance * teamShare) / SHAREDIVISOR
812   );
813
814   emit SwapAndSendFee(tokenAmount, newBalance);
815
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 822

## low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

## Source File

- KISEKI.sol

## Locations

```
821   require(
822   newTeamShare + newMarketingShare == SHAREDIVISOR,
823   "Sum of shares must be 100"
824   );
825
826
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 791

## low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

## Source File

- KISEKI.sol

## Locations

```
790   address[] memory path = new address[](2);
791   path[0] = address(this);
792   path[1] = uniswapV2Router.WETH();
793
794   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
795
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 792

## low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

## Source File

- KISEKI.sol

## Locations

```
791   path[0] = address(this);
792   path[1] = uniswapV2Router.WETH();
793
794   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
795   tokenAmount,
796
```

# DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Sysfixed's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of 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. In order 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 on the basis of 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 below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

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 Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

# ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.