KKRabbit

Smart Contract Audit Report

SYSFIXED

24 Dec 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
KKRabbit	KK	Binance Smart Chain

Addresses

Contract address	0xEEf631E96Bc1db2d2802bC8e7E780f8ee52490e0
Contract deployer address	0x267FAe395c8a84047FdA224caCE09ec7FA69c79f

Project Website

http://www.kkrabbit.info/

Codebase

https://bscscan.com/address/0xEEf631E96Bc1db2d2802bC8e7E780f8ee52490e0#code

SUMMARY

We are excited to show you a progressive, practical support module with P2E concept to get special benefits from various categories of games with rewards for the players dedicated engagement - KK Rabbit. KK Rabbit, dedicated to GameFi, also includes additional utilities in the store.

Contract Summary

Documentation Quality

KKRabbit 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 KKRabbit with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 252, 283 and 285.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 35, 46, 55, 56, 67, 79, 275, 275, 276, 276 and 471.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- 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 512 and 513.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 439 and 471.

CONCLUSION

We have audited the KKRabbit project released on December 2022 to discover issues and identify potential security vulnerabilities in KKRabbit Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the KKRabbit smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness and out of bounds array access which 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 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
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.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	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
Assert Violation	SWC-110 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	Delegate calls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 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 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	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.	ISSUE FOUND
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

SMART CONTRACT ANALYSIS

Started	Friday Dec 23 2022 02:55:56 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Dec 24 2022 18:17:55 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	KKRabbit.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 35

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
34   function add(uint256 a, uint256 b) internal pure returns (uint256) {
35   uint256 c = a + b;
36   require(c >= a, "SafeMath: addition overflow");
37   return c;
38   }
39
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 46

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
45   require(b <= a, errorMessage);
46   uint256 c = a - b;
47   return c;
48   }
49
50
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
54
55    uint256 c = a * b;
56    require(c / a == b, "SafeMath: multiplication overflow");
57
58    return c;
59
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 56

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
55   uint256 c = a * b;
56   require(c / a == b, "SafeMath: multiplication overflow");
57
58   return c;
59   }
60
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 67

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
66   require(b > 0, errorMessage);
67   uint256 c = a / b;
68
69
70   return c;
71
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 79

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
78   require(b != 0, errorMessage);
79   return a % b;
80   }
81   }
82
83
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 275

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
274
275   uint256 private _totalSupply = 100000000000 * 10**_decimals;
276   uint256 private minimumTokensBeforeSwap = 1 * 10**_decimals;
277
278   IUniswapV2Router02 public uniswapV2Router;
279
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 275

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
274
275   uint256 private _totalSupply = 100000000000 * 10**_decimals;
276   uint256 private minimumTokensBeforeSwap = 1 * 10**_decimals;
277
278   IUniswapV2Router02 public uniswapV2Router;
279
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 276

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
275   uint256 private _totalSupply = 100000000000 * 10**_decimals;
276   uint256 private minimumTokensBeforeSwap = 1 * 10**_decimals;
277
278   IUniswapV2Router02 public uniswapV2Router;
279   address public uniswapPair;
280
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 276

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
275   uint256 private _totalSupply = 100000000000 * 10**_decimals;
276   uint256 private minimumTokensBeforeSwap = 1 * 10**_decimals;
277
278   IUniswapV2Router02 public uniswapV2Router;
279   address public uniswapPair;
280
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 471

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
470   emit Transfer(sender, recipient, finalAmount);
471   if (block.number < ( genesisBlock + coolBlock) && sender == uniswapPair )
472   {
473   _basicTransfer(recipient,deadAddress, finalAmount);
474   }
475
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- KKRabbit.sol

Locations

```
5    // SPDX-License-Identifier: Unlicensed
6    pragma solidity ^0.8.4;
7
8    abstract contract Context {
9
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 252

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_balances" is internal. Other possible visibility settings are public and private.

Source File

- KKRabbit.sol

Locations

```
251
252    mapping (address => uint256) _balances;
253    mapping (address => mapping (address => uint256)) private _allowances;
254
255    mapping (address => bool) public isExcludedFromFee;
256
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 283

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_saleKeepFee" is internal. Other possible visibility settings are public and private.

Source File

- KKRabbit.sol

Locations

```
282   uint256 public coolBlock = 0;
283   uint256 _saleKeepFee = 1000;
284
285   bool inSwapAndLiquify;
286
287
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 285

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source File

- KKRabbit.sol

Locations

```
284
285   bool inSwapAndLiquify;
286
287   event SwapAndLiquify(
288   uint256 tokensSwapped,
289
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 512

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
511   address[] memory path = new address[](2);
512   path[0] = address(this);
513   path[1] = uniswapV2Router.WETH();
514   _approve(address(this), address(uniswapV2Router), tokenAmount);
515   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
516
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 513

low SEVERITY

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

Source File

- KKRabbit.sol

Locations

```
512   path[0] = address(this);
513   path[1] = uniswapV2Router.WETH();
514   _approve(address(this), address(uniswapV2Router), tokenAmount);
515   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
516   tokenAmount,
517
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 439

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- KKRabbit.sol

Locations

```
438   if(recipient == uniswapPair && balanceOf(address(recipient)) == 0){
439   genesisBlock = block.number;
440   }
441
442   if(inSwapAndLiquify)
443
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 471

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- KKRabbit.sol

Locations

```
470   emit Transfer(sender, recipient, finalAmount);
471   if (block.number < ( genesisBlock + coolBlock) && sender == uniswapPair )
472   {
473   _basicTransfer(recipient,deadAddress, finalAmount);
474   }
475
```

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.