



CryptoBunnyClub  
Smart Contract  
Audit Report

# TABLE OF CONTENTS

## [Audited Details](#)

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

## [Summary](#)

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

## [Conclusion](#)

## [Audit Results](#)

## [Smart Contract Analysis](#)

- Detected Vulnerabilities

## [Disclaimer](#)

## [About Us](#)

# AUDITED DETAILS

## Audited Project

Project name	Token ticker	Blockchain
CryptoBunnyClub	CBC	BSC

## Addresses

Contract address	0xF92164dbd3E8D80655124f22f68C9337321F227f
Contract deployer address	0x1d875Aefae30c30F4A152cdc99b78a402177899E

## Project Website

<https://cb-club.org/>

## Codebase

<https://bscscan.com/address/0xF92164dbd3E8D80655124f22f68C9337321F227f#code>

# SUMMARY

The Cryptobunnyclub starts with a low MC , website + own NFT mining page, play 2 earn a game in progress, for pc, mobile, Xbox, ps4, mint NFT + huge marketing, Tokenomics details, 2% BUSD rewards 2% lp 1% marketing, jump & run, battle royale, street fight, battle royal function, where 100 bunnies can battle against each other solo, co-op or as a team of 4, bunnys street fight where you can use tokens and face each other in a 1-vs-1 fight and much more.

## Contract Summary

### Documentation Quality

Cryptobunnyclub provides a document with a very good standard of solidity base code.

- The technical description is provided clearly and structured and also don't have any risk issue.

### Code Quality

The Overall quality of the basecode is GOOD

- Standart solidity basecode and rules are already followed with Cryptobunnyclub Project .

### Test Coverage

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

## Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 29, 31, 36, 39, 44, 56, 65, 72, 73, 81, 200, 370, 383, 426, 485, 491, 555, 610, 610, 758, 947, 947, 1090, 1100, 1104, and 200.
- SWC-103 | A floating pragma is set on lines 6. The current pragma Solidity directive is ""^0.8.17"". 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.
- SWC-108 | State variable visibility is not set on lines 492.It is best practice to set the visibility of state variables explicitly. The default visibility for "protections" is internal. Other possible visibility settings are public and private.
- SWC-110 | Out of bounds array access on lines 171, 201, 206, 869, 870, 871, 885, 886, and 1096.

## CONCLUSION

We have audited the CryptoBunnyClub Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in CryptoBunnyClub 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 produced satisfactory results with a low risk issue on the contract project.

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 that we found were assert violation, floating pragma set, and default visibility. The functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.

# 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
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	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 Caller	SWC-112	Delegatecalls 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.	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

# SMART CONTRACT ANALYSIS

Started	Sat Jan 14 2023 04:10:17 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 15 2023 05:12:27 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Bunny.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-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged



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

LINE 29

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
28 function mul(int256 a, int256 b) internal pure returns (int256) {
29   int256 c = a * b;
30   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
31   require((b == 0) || (c / b == a));
32   return c;
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 31

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
30  require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
31  require((b == 0) || (c / b == a));
32  return c;
33  }
34  function div(int256 a, int256 b) internal pure returns (int256) {
```

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

LINE 36

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
35  require(b != -1 || a != MIN_INT256);
36  return a / b;
37  }
38  function sub(int256 a, int256 b) internal pure returns (int256) {
39  int256 c = a - b;
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 39

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
38 function sub(int256 a, int256 b) internal pure returns (int256) {
39     int256 c = a - b;
40     require((b >= 0 && c <= a) || (b < 0 && c > a));
41     return c;
42 }
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 44

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
43  function add(int256 a, int256 b) internal pure returns (int256) {
44  int256 c = a + b;
45  require((b >= 0 && c >= a) || (b < 0 && c < a));
46  return c;
47  }
```

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

LINE 56

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
55  function add(uint256 a, uint256 b) internal pure returns (uint256) {
56  uint256 c = a + b;
57  require(c >= a, "SafeMath: addition overflow");
58  return c;
59  }
```



# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 65

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
64  require(b <= a, errorMessage);
65  uint256 c = a - b;
66  return c;
67  }
68  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
```

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

LINE 72

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
71  }  
72  uint256 c = a * b;  
73  require(c / a == b, "SafeMath: multiplication overflow");  
74  return c;  
75  }
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 73

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
72  uint256 c = a * b;
73  require(c / a == b, "SafeMath: multiplication overflow");
74  return c;
75  }
76  function div(uint256 a, uint256 b) internal pure returns (uint256) {
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 81

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
80  require(b > 0, errorMessage);  
81  uint256 c = a / b;  
82  return c;  
83  }  
84
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 200

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
199  uint index = map.indexOf[key];
200  uint lastIndex = map.keys.length - 1;
201  address lastKey = map.keys[lastIndex];
202
203  map.indexOf[lastKey] = index;
```

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

LINE 370

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
369 // see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
370 uint256 constant internal magnitude = 2**128;
371 uint256 internal magnifiedDividendPerShare;
372 mapping(address => int256) internal magnifiedDividendCorrections;
373 mapping(address => uint256) internal withdrawnDividends;
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 383

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
382 magnifiedDividendPerShare = magnifiedDividendPerShare.add(  
383 (amount).mul(magnitude) / totalSupply()  
384 );  
385 emit DividendsDistributed(msg.sender, amount);  
386
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 426

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
425     function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
426     return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
427     .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
428     }
429 }
```



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

LINE 485

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
484 uint256 public _previousTotalFees = totalFees;
485 uint256 public swapTokensAtAmount = 100000 * (10**18);
486 address public _marketingWalletAddress =
0x3785a6fCC98dA2a768646403fe8E2671C98bB94F;
487 uint256 public gasForProcessing = 300000;
488
```

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

LINE 491

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
490 uint256 public maxTransferAmountRate = 300; //divisor 10000 => for 3%
491 uint256 public _maxWalletBalance = 3000000 * 10 ** 18;
492 mapping(address => bool) excludedFromAntiWhale;
493 mapping(address => bool) private _isExcludedFromMaxWallet;
494 mapping (address => bool) public automatedMarketMakerPairs;
```

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

LINE 555

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
554
555  _mint(_msgSender(), 100000000 * (10**18));
556
557  }
558
```

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

LINE 610

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
609     function setMaxBalance(uint256 maxBalancePercent) external onlyOwner {
610         _maxWalletBalance = maxBalancePercent * 10 ** 18;
611     }
612
613     function includeAndExcludedFromMaxWallet(address account, bool value) public
onlyOwner {
```

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

LINE 610

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
609     function setMaxBalance(uint256 maxBalancePercent) external onlyOwner {
610         _maxWalletBalance = maxBalancePercent * 10 ** 18;
611     }
612
613     function includeAndExcludedFromMaxWallet(address account, bool value) public
onlyOwner {
```

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

LINE 758

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
757     uint256 currentBalance = balanceOf(to);
758     require(!_isExcludedFromMaxWallet[to] || (currentBalance + amount <=
_maxWalletBalance),
759     "ERC20: Reached max wallet holding");
760 }
761
```

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

LINE 947

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
946   claimWait = 3600; // one hour
947   minimumTokenBalanceForDividends = 1 * (10**18); //must hold 1+ tokens
948   }
949
950   function _transfer(address, address, uint256) internal override {
```

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

LINE 947

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
946   claimWait = 3600; // one hour
947   minimumTokenBalanceForDividends = 1 * (10**18); //must hold 1+ tokens
948   }
949
950   function _transfer(address, address, uint256) internal override {
```



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

LINE 1090

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
1089 while(gasUsed < gas && iterations < numberOfTokenHolders) {  
1090   _lastProcessedIndex++;  
1091  
1092   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {  
1093     _lastProcessedIndex = 0;
```

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

LINE 1100

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
1099   if(processAccount(payable(account), true)) {  
1100     claims++;  
1101   }  
1102   }  
1103
```

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

LINE 1104

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
1103
1104  iterations++;
1105
1106  uint256 newGasLeft = gasleft();
1107
```

# SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 200

## low SEVERITY

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

## Source File

- Bunny.Sol

## Locations

```
199 uint index = map.indexOf[key];
200 uint lastIndex = map.keys.length - 1;
201 address lastKey = map.keys[lastIndex];
202
203 map.indexOf[lastKey] = index;
```

## SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

### low SEVERITY

The current pragma Solidity directive is `""^0.6.12""`. 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

- Bunny.Sol

### Locations

```
5 // SPDX-License-Identifier: Unlicensed
6 pragma solidity ^0.6.12;
7
8 abstract contract Context {
9     function _msgSender() internal view virtual returns (address) {
```

## SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 492

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
491 uint256 public _maxWalletBalance = 3000000 * 10 ** 18;
492 mapping(address => bool) excludedFromAntiWhale;
493 mapping(address => bool) private _isExcludedFromMaxWallet;
494 mapping (address => bool) public automatedMarketMakerPairs;
495
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 171

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
170     function getKeyAtIndex(Map storage map, uint index) public view returns (address) {  
171         return map.keys[index];  
172     }  
173  
174
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 201

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
200 uint lastIndex = map.keys.length - 1;
201 address lastKey = map.keys[lastIndex];
202
203 map.indexOf[lastKey] = index;
204 delete map.indexOf[key];
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 206

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
205
206   map.keys[index] = lastKey;
207   map.keys.pop();
208   }
209   }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 869

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
868     address[] memory path = new address[](3);
869     path[0] = address(this);
870     path[1] = uniswapV2Router.WETH();
871     path[2] = BUSD;
872     _approve(address(this), address(uniswapV2Router), tokenAmount);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 870

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
869 path[0] = address(this);
870 path[1] = uniswapV2Router.WETH();
871 path[2] = BUSD;
872 _approve(address(this), address(uniswapV2Router), tokenAmount);
873 uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 871

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
870 path[1] = uniswapV2Router.WETH();
871 path[2] = BUSD;
872 _approve(address(this), address(uniswapV2Router), tokenAmount);
873 uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
874 tokenAmount,
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 885

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
884   require(path.length <= 2, "fail");
885   path[0] = address(this);
886   path[1] = uniswapV2Router.WETH();
887   _approve(address(this), address(uniswapV2Router), tokenAmount);
888   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 886

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
885 path[0] = address(this);
886 path[1] = uniswapV2Router.WETH();
887 _approve(address(this), address(uniswapV2Router), tokenAmount);
888 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
889 tokenAmount,
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1096

### low SEVERITY

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

### Source File

- Bunny.Sol

### Locations

```
1095
1096 address account = tokenHoldersMap.keys[_lastProcessedIndex];
1097
1098 if(canAutoClaim(lastClaimTimes[account])) {
1099 if(processAccount payable(account), true) {
```

# 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.