Goge

Smart Contract Audit Report

SYSFIXED

25 Jan 2023

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
Goge	GOG	BSC

Addresses

Contract address	0x7154Ea33E38E0f21917956007590A4997b34a105
Contract deployer address	0x6334BAE02114C080F05E6D58b65A1d7926FbbeBc

Project Website

https://goge.io/

Codebase

https://bscscan.com/address/0x7154Ea33E38E0f21917956007590A4997b34a105#code

SUMMARY

A "non-fungible token" set of 10K Goge on the BSC blockchain is offered in several sets. A playful ball-like character that will introduce you to the memories and real history of World Cups in NFT format and remind you of past moments, each holder has private and exclusive access to channels, events, individual IP opportunities, whitelists, NFT and receive benefits across the Goge ecosystem. benefits among them are minting live on the website, direct buying or selling on opensea, NFTs staking and KYC or audit.

Contract Summary

Documentation Quality

The Goge project provides a good document and a good contract with standard solidity written contract.

- There is a lot of technical description provided by Goge Project

Code Quality

Overall quality code is good and well structured

- The official Solidity codes guide is followed.

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 119, 299, 325, 356, 378, 385, 397, 398, 400, 401, 402, 478, 534, 535, 546, 547, 548, 562, 564, 588, 590, 594,
- 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-103 | State variable visibility is not set on lines 105, 146, 155 .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-115 | Use of "tx.origin" as a part of authorization control on lines 439. The index access expression can cause an exception in case an invalid array index value is used.
- SWC-110 | Out of bounds array access on lines 494, 495, 547, 548
- SWC-120 | OPotential use of "block.number" as source of randonmness on lines 531

CONCLUSION

CONCLUSION

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

The security audit report produced satisfactory results with a low risk issue on 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 were found were assert violation, a floating pragma is setn and weak sources of the randomness contained in the contract

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.	ISSUE FOUND
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	Mon Jan 23 2023 05:34:23 GMT+0000 (Coordinated Universal Time)
Finished	Tu Jan 24 2023 07:09:06 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Goge.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-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-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	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-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 119

low SEVERITY

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

Source File

- Goge.sol

Locations

```
118   uint8 constant private _decimals = 9;
119   uint256 constant private _tTotal = startingSupply * 10**_decimals;
120   struct Fees {
121   |
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 299

low SEVERITY

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

Source File

- Goge.sol

Locations

```
298   if (_allowances[sender][msg.sender] != type(uint256).max) {
299   _allowances[sender][msg.sender] -= amount;
300   }
301   |
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 325

low SEVERITY

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

Source File

- Goge.sol

Locations

```
324   if (timeSinceLastPair != 0) {
325   require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
326   }
327   lpPairs[pair] = true;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 356

low SEVERITY

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

Source File

- Goge.sol

Locations

```
355   function getCirculatingSupply() public view returns (uint256) {
356   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
357   }
358   |
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- Goge.sol

Locations

```
377   "Cannot exceed maximums.");
378   require(buyFee + sellFee <= maxRoundtripTax, "Cannot exceed roundtrip maximum.");
379   _taxRates.buyFee = buyFee;
380   _taxRates.sellFee = sellFee;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 385

low SEVERITY

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

Source File

- Goge.sol

Locations

```
384   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
385   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
386   }
387   |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- Goge.sol

Locations

```
396   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
397   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
398   swapAmount = (_tTotal * amountPercent) / amountDivisor;
399   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 398

low SEVERITY

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

Source File

- Goge.sol

Locations

```
397   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
398   swapAmount = (_tTotal * amountPercent) / amountDivisor;
399   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
400   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 400

low SEVERITY

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

Source File

- Goge.sol

Locations

```
399   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
400   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
401   require(swapAmount >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
402   require(swapThreshold >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 401

low SEVERITY

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

Source File

- Goge.sol

Locations

```
400   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
401   require(swapAmount >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
402   require(swapThreshold >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
403   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- Goge.sol

Locations

```
401   require(swapAmount >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
402   require(swapThreshold >= _tTotal / 1000000, "Cannot be lower than 0.00001% of total
supply.");
403   }
404   |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 478

low SEVERITY

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

Source File

- Goge.sol

Locations

```
477   uint256 swapAmt = swapAmount;
478   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
479   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
480   contractSwap(contractTokenBalance);
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 534

low SEVERITY

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

Source File

- Goge.sol

Locations

```
533   allowedPresaleExclusion = false;
534   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
535   swapAmount = (balanceOf(lpPair) * 30) / 10000;
536   launchStamp = block.timestamp;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 535

low SEVERITY

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

Source File

- Goge.sol

Locations

```
534   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
535   swapAmount = (balanceOf(lpPair) * 30) / 10000;
536   launchStamp = block.timestamp;
537   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 546

low SEVERITY

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

Source File

- Goge.sol

Locations

```
545   require(accounts.length == amounts.length, "Lengths do not match.");
546   for (uint16 i = 0; i < accounts.length; i++) {
547   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
548   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 547

low SEVERITY

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

Source File

- Goge.sol

Locations

```
546   for (uint16 i = 0; i < accounts.length; i++) {
547   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
548   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
549   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 548

low SEVERITY

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

Source File

- Goge.sol

Locations

```
547   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
548   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
549   }
550   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 562

low SEVERITY

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

Source File

- Goge.sol

Locations

```
561   }
562   _tOwned[from] -= amount;
563   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
564   _tOwned[to] += amountReceived;
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 564

low SEVERITY

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

Source File

- Goge.sol

Locations

```
563   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
564   _tOwned[to] += amountReceived;
565   emit Transfer(from, to, amountReceived);
566   if (!_hasLiqBeenAdded) {
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- Goge.sol

Locations

```
587   || block.chainid == 56)) { currentFee = protectionValue; }
588   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
589   if (feeAmount > 0) {
590   _tOwned[address(this)] += feeAmount;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 590

low SEVERITY

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

Source File

- Goge.sol

Locations

```
589   if (feeAmount > 0) {
590   _tOwned[address(this)] += feeAmount;
591   emit Transfer(from, address(this), feeAmount);
592   }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 594

low SEVERITY

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

Source File

- Goge.sol

Locations

```
593   }
594   return amount - feeAmount;
595   }
596   }
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.9.0"". 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

- Goge.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity >=0.6.0 <0.9.0;
7   interface IERC20 {
8   |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 105

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "lpPairs" is internal. Other possible visibility settings are public and private. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Goge.sol

Locations

```
104   mapping (address => uint256) private _tOwned;
105   mapping (address => bool) lpPairs;
106   uint256 private timeSinceLastPair = 0;
107   mapping (address => mapping (address => uint256)) private _allowances;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 146

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "lpPairs" is internal. Other possible visibility settings are public and private. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Goge.sol

Locations

```
145   address public marketingWallet =
payable(0x9f63626EDc27680f9Fd166BE829Ed97f2f116388);
146   bool inSwap;
147   bool public contractSwapEnabled = false;
148   uint256 public swapThreshold;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 155

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "lpPairs" is internal. Other possible visibility settings are public and private. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Goge.sol

Locations

```
154   bool public _hasLiqBeenAdded = false;
155   Protections protections;
156   uint256 public launchStamp;
157   |
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 439

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- Goge.sol

Locations

```
438   && to != _owner
439   && tx.origin != _owner
440   && !_liquidityHolders[to]
441   && !_liquidityHolders[from]
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 494

low SEVERITY

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

Source File

- Goge.sol

Locations

```
493   address[] memory path = new address[](2);
494   path[0] = address(this);
495   path[1] = dexRouter.WETH();
496   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 495

low SEVERITY

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

Source File

- Goge.sol

Locations

```
494   path[0] = address(this);
495   path[1] = dexRouter.WETH();
496   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
497   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 547

low SEVERITY

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

Source File

- Goge.sol

Locations

```
546   for (uint16 i = 0; i < accounts.length; i++) {
547   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
548   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
549   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 548

low SEVERITY

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

Source File

- Goge.sol

Locations

```
547   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
548   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
549   }
550   }
```

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

LINE 531

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

- Goge.sol

Locations

```
530   }
531   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
532   tradingEnabled = true;
533   allowedPresaleExclusion = false;
```

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.