Cheems Token

Smart Contract Audit Report

SYSFIXED

16 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Cheems Token	CHEEMS	BSC

Addresses

Contract address	0x299F8fBD8Eb6877dC6AeF79C9754fD7fF548D280
Contract deployer address	0xB0004dF5aBcf68D2b0Ae6F1445cda84c606Ae591

Project Website

https://www.cheemstoken.com/

Codebase

https://bscscan.com/address/0x299F8fBD8Eb6877dC6AeF79C9754fD7fF548D280#code

SUMMARY

The new dog king is in a building called Cheems. The first bsc dog coin for the people, by the people with 50% of the total supply is burned. The cheems team was tired of toxic “fud binance” and wanted to make a fun memecoin where everyone gets a fair shot. In-built staking rewards system for all its holders. The advantage is an audit, 5% auto-reflection to holders per buy/sell, nft alpha of, cmc + cg fast track, no unlocked tokens, by ex-sol dev, and community driven.

Contract Summary

Documentation Quality

Cheems Token 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 high risk issue.

Code Quality

The Overall quality of the basecode is GOOD

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

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 411, 438, 471, 474, 490, 519, 521, 574, 697, 712, 729, 730, 744, 755, 766, 778, 789, 796, 821, 837, 855, 869, 1299, 1301, 1318, 1394, 1395, 1471, 1502, 1507, 1511, and 1395.
- SWC-108 | State variable visibility is not set on lines 1251.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 1394, 1395, 1395, 1472, dan 1476.

CONCLUSION

We have audited the Cheems Token Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in Cheems Token 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 were found stated variable visibility is not set, and a floating pragma is set. We are commended 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.

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.	PASS
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	Sun Jan 15 2023 05:46:31 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jan 16 2023 06:40:31 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	cheems.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	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-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-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- cheems.sol

Locations

```
410   require(account != address(0), "ERC20: mint to the zero address");
411   _beforeTokenTransfer(address(0), account, amount);
412   _totalSupply += amount;
413   |
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 438

low SEVERITY

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

Source File

- cheems.sol

Locations

```
437   require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
438   unchecked {
439   _balances[account] = accountBalance - amount;
440   }
441   _totalSupply -= amount;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 471

low SEVERITY

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

Source File

- cheems.sol

Locations

```
470   emit Approval(owner, spender, amount);
471   }
472   /**
473   * @dev Updates `owner` s allowance for `spender` based on spent `amount`.
474   *
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 474

low SEVERITY

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

Source File

- cheems.sol

Locations

```
473   /**
474   * @dev Updates `owner` s allowance for `spender` based on spent `amount`.
475   *
476   * Does not update the allowance amount in case of infinite allowance.
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 490

low SEVERITY

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

Source File

- cheems.sol

Locations

```
489   unchecked {
490   _approve(owner, spender, currentAllowance - amount);
491   }
492   }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 519

low SEVERITY

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

Source File

- cheems.sol

Locations

```
518   * minting and burning.
519   *
520   * Calling conditions:
521   *
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 521

low SEVERITY

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

Source File

- cheems.sol

Locations

```
520   *
521   * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
522   * has been transferred to `to`.
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 574

low SEVERITY

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

Source File

- cheems.sol

Locations

```
573   *
574   * NOTE: Renouncing ownership will leave the contract without an owner,
575   * thereby removing any functionality that is only available to the owner.
576   */
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 697

low SEVERITY

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

Source File

- cheems.sol

Locations

```
696   /**
697   * @dev Returns the multiplication of two unsigned integers, reverting on
698   * overflow.
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 712

low SEVERITY

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

Source File

- cheems.sol

Locations

```
711   * @dev Returns the integer division of two unsigned integers, reverting on
712   * division by zero. The result is rounded towards zero.
713   *
714   * Counterpart to Solidity's `/` operator.
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 729

low SEVERITY

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

Source File

- cheems.sol

Locations

```
728   * Counterpart to Solidity's `%` operator. This function uses a `revert`
729   * opcode (which leaves remaining gas untouched) while Solidity uses an
730   * invalid opcode to revert (consuming all remaining gas).
731   *
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 730

low SEVERITY

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

Source File

- cheems.sol

Locations

```
729   * opcode (which leaves remaining gas untouched) while Solidity uses an
730   * invalid opcode to revert (consuming all remaining gas).
731   *
732   * Requirements:
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 744

low SEVERITY

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

Source File

- cheems.sol

Locations

```
743   * overflow (when the result is negative).
744   *
745   * CAUTION: This function is deprecated because it requires allocating memory for
the error
746   * CAUTION: This function is deprecated because it requires allocating memory for
the error
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 755

low SEVERITY

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

Source File

- cheems.sol

Locations

```
754   uint256 a,
755   uint256 b,
756   string memory errorMessage
757   ) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 766

low SEVERITY

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

Source File

- cheems.sol

Locations

```
765   * @dev Returns the integer division of two unsigned integers, reverting with custom
message on
766   * division by zero. The result is rounded towards zero.
767   *
768   * Counterpart to Solidity's `/` operator. Note: this function uses a
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 778

low SEVERITY

Arithmetic operation "-" discovered

Source File

- cheems.sol

Locations

```
777   uint256 a,
778   uint256 b,
779   string memory errorMessage
780   ) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 789

low SEVERITY

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

Source File

- cheems.sol

Locations

```
788   * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
modulo),
789   * reverting with custom message when dividing by zero.
790   *
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 796

low SEVERITY

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

Source File

- cheems.sol

Locations

```
795   * opcode (which leaves remaining gas untouched) while Solidity uses an
796   * invalid opcode to revert (consuming all remaining gas).
797   *
798   * Requirements:
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 821

low SEVERITY

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

Source File

- cheems.sol

Locations

```
820   * It is unsafe to assume that an address for which this function returns
821   * false is an externally-owned account (EOA) and not a contract.
822   *
823   * Among others, `isContract` will return false for the following
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 837

low SEVERITY

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

Source File

- cheems.sol

Locations

```
836   * Preventing calls from contracts is highly discouraged. It breaks composability,
breaks support for smart wallets
837   * like Gnosis Safe, and does not provide security since it can be circumvented by
calling from a contract
838   * constructor.
839   * ====
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 855

low SEVERITY

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

Source File

- cheems.sol

Locations

```
854   * of certain opcodes, possibly making contracts go over the 2300 gas limit
855   * imposed by `transfer`, making them unable to receive funds via
856   * imposed by `transfer`, making them unable to receive funds via
857   * `transfer`. {sendValue} removes this limitation.
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 869

low SEVERITY

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

Source File

- cheems.sol

Locations

```
868   (bool success, ) = recipient.call{value: amount}("");
869   require(success, "Address: unable to send value, recipient may have reverted");
870   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1299

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1298   developmentWalletAddress = account;
1299   _isExcludedFromFee[account] = true;
1300   }
1301   function setMarketingAddress(address account) public onlyOwner() {
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1301

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1300   _isExcludedFromFee[account] = true;
1301   }
1302   function setMarketingAddress(address account) public onlyOwner() {
1303   _marketingWalletAddress = account;
1304   _isExcludedFromFee[account] = true;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1318

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1317   }
1318   function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view
returns(uint256) {
1319   require(tAmount <= _tTotal, "Amount must be less than supply");
1320   if (!deductTransferFee) {
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1394

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1393   }
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1395

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
1397   uint256 rDevelopment = tfees.development.mul(currentRate);
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1471

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1470   require(from != address(0), "ERC20: transfer from the zero address");
1471   require(to != address(0), "ERC20: transfer to the zero address");
1472   require(amount > 0, "Transfer amount must be greater than zero");
1473   |
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1502

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1501   _transferStandard(sender, recipient, amount);
1502   } else if (_isExcluded[sender] && _isExcluded[recipient]) {
1503   _transferBothExcluded(sender, recipient, amount);
1504   } else {
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1507

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1506   }
1507   if(!takeFee)
1508   restoreAllFee();
1509   }
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1511

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1510   function _transferStandard(address sender, address recipient, uint256 tAmount)
private {
1511   (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount,
uint256 tFee, uint256 tDevelopment, uint256 tMarketing) = _getValues(tAmount);
1512   _rOwned[sender] = _rOwned[sender].sub(rAmount);
1513   _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
```

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

LINE 1395

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
1397   uint256 rDevelopment = tfees.development.mul(currentRate);
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1251

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

- cheems.sol

Locations

```
1250   address public immutable uniswapV2Pair;
1251   bool inSwapAndLiquify;
1252   bool public swapAndLiquifyEnabled = true;
1253   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1394

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1393   }
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1395

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
1397   uint256 rDevelopment = tfees.development.mul(currentRate);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1395

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1394   function _getRValues(uint256 tAmount, TFees memory tfees, uint256 currentRate)
private pure returns (uint256, uint256, uint256) {
1395   uint256 rAmount = tAmount.mul(currentRate);
1396   uint256 rFee = tfees.tax.mul(currentRate);
1397   uint256 rDevelopment = tfees.development.mul(currentRate);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1472

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1471   require(to != address(0), "ERC20: transfer to the zero address");
1472   require(amount > 0, "Transfer amount must be greater than zero");
1473   uint256 contractTokenBalance = balanceOf(address(this));
1474   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1476

low SEVERITY

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

Source File

- cheems.sol

Locations

```
1475   uint256 contractTokenBalance = balanceOf(address(this));
1476   bool overMinTokenBalance = contractTokenBalance >= numTokensSellToAddToLiquidity;
1477   if (
1478   overMinTokenBalance &&
1479   |
```

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.