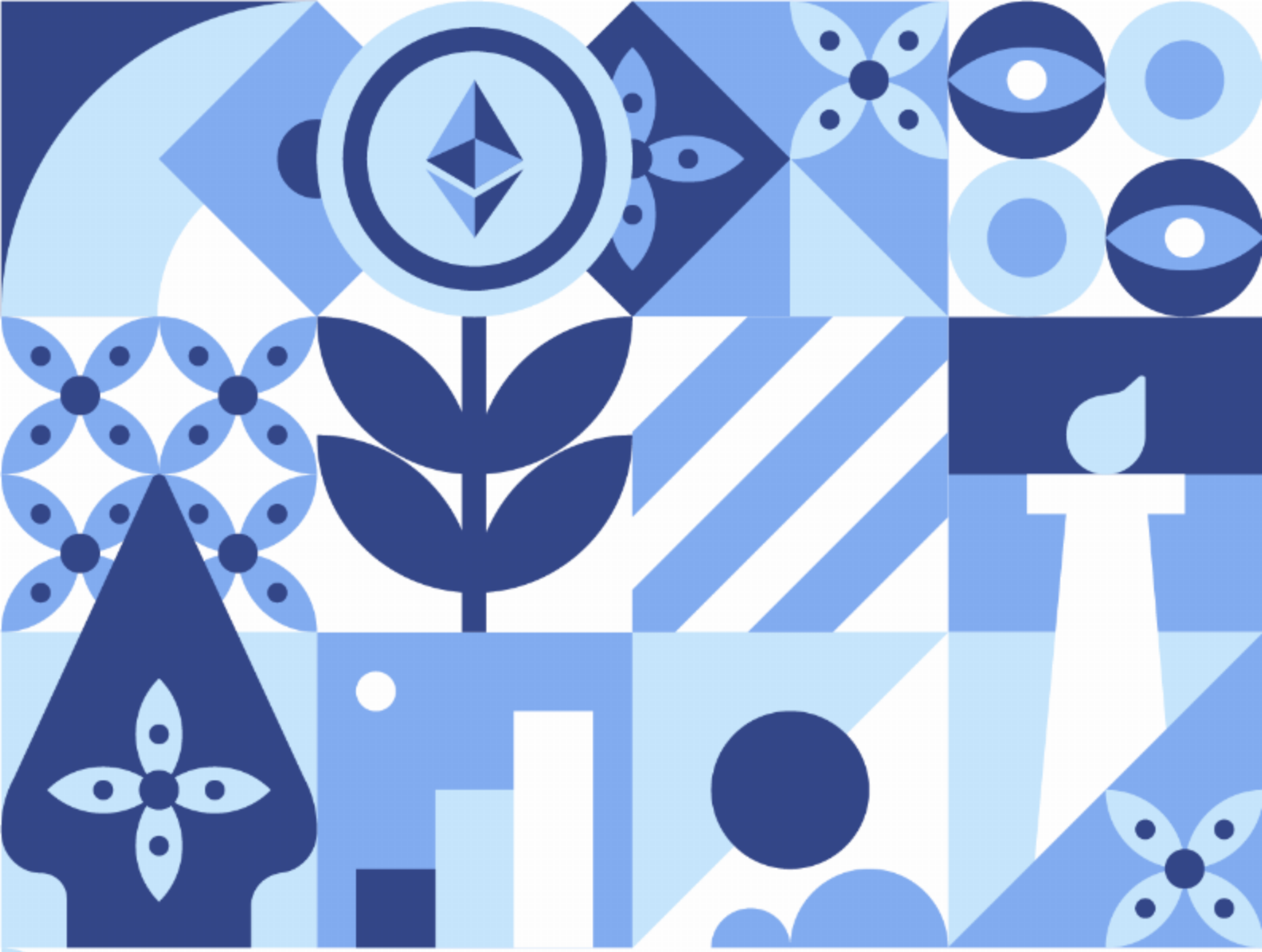

Chronoly

Smart Contract Audit Report

SYSFIXED

26 Apr 2022

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
Chronoly	CRNO	Ethereum

Addresses

Contract address	0xE23311294467654E0CaB14cD32A169A41be5ca8E
Contract deployer address	0xE23311294467654E0CaB14cD32A169A41be5ca8E

Project Website

https://chronoly.io/

Codebase

https://etherscan.io/address/0xE23311294467654E0CaB14cD32A169A41be5ca8E#code

SUMMARY

Chronoly is the world's first 24/7 watch investment platform that is making it possible for anyone to fractionally invest in rare and exclusive timepieces. Each NFT watch that we mint is backed by the physical version of the watch. The watch NFT is then broken down into fractions making it easy for anyone to invest in the watch from as little as $10.

Contract Summary

Documentation Quality

Chronoly 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 Chronoly 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 438.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 39, 55, 58, 58, 78, 426, 426, 453, 453, 456, 457, 472, 473, 678, 678, 678, 678, 678, 682, 689, 692, 692 and 692.
- 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 423, 674, 743 and 744.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 691.

CONCLUSION

We have audited the Chronoly project released on April 2022 to discover issues and identify potential security vulnerabilities in Chronoly 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 Chronoly 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 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. We recommend setting the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private also don't use any of those environment variables as sources of randomness and be aware that the use of these variables introduces a certain level of trust in miners.

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
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 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
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Monday Apr 25 2022 12:41:45 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Apr 26 2022 12:11:05 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	CRNOTOKEN.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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	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 39

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
38   function sub(uint a, uint b, string memory errorMessage) internal pure returns
(uint) {
39   require(b <= a, errorMessage);
40   uint c = a - b;
41
42   return c;
43
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
54   function div(uint a, uint b) internal pure returns (uint) {
55   return div(a, b, "SafeMath: division by zero");
56   }
57   function div(uint a, uint b, string memory errorMessage) internal pure returns
(uint) {
58   // Solidity only automatically asserts when dividing by 0
59
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 58

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
57   function div(uint a, uint b, string memory errorMessage) internal pure returns
(uint) {
58   // Solidity only automatically asserts when dividing by 0
59   require(b > 0, errorMessage);
60   uint c = a / b;
61
62
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 58

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
57   function div(uint a, uint b, string memory errorMessage) internal pure returns
(uint) {
58   // Solidity only automatically asserts when dividing by 0
59   require(b > 0, errorMessage);
60   uint c = a / b;
61
62
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 78

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
77
78   event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
79
80   /**
81   * @dev Initializes the contract setting the deployer as the initial owner.
82
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 426

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
425   uint256 public BUYliquidityFee = 1;
426   uint256 public BUYtotalFee =
BUYliquidityFee.add(BUYmarketingFee).add(BUYrewardFee).add(BUYburnFee);
427
428   uint256 public SELLmarketingFee = 6;
429   uint256 public SELLrewardFee = 3;
430
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 426

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
425   uint256 public BUYliquidityFee = 1;
426   uint256 public BUYtotalFee =
BUYliquidityFee.add(BUYmarketingFee).add(BUYrewardFee).add(BUYburnFee);
427
428   uint256 public SELLmarketingFee = 6;
429   uint256 public SELLrewardFee = 3;
430
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 453

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
452
453   bool private swapping;
454
455
456   modifier lockTheSwap {
457
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 453

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
452
453   bool private swapping;
454
455
456   modifier lockTheSwap {
457
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 456

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
455
456   modifier lockTheSwap {
457   inSwapAndLiquify = true;
458   _;
459   inSwapAndLiquify = false;
460
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 457

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
456   modifier lockTheSwap {
457   inSwapAndLiquify = true;
458   _;
459   inSwapAndLiquify = false;
460   }
461
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 472

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
471   // Create a uniswap pair for this new token
472   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
473   .createPair(address(this), _uniswapV2Router.WETH());
474
475   // set the rest of the contract variables
476
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
472   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
473   .createPair(address(this), _uniswapV2Router.WETH());
474
475   // set the rest of the contract variables
476   uniswapV2Router = _uniswapV2Router;
477
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 678

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
677
678   return (seed - ((seed / lotteryEligibles.length) * lotteryEligibles.length));
679
680   }
681
682
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 678

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
677
678   return (seed - ((seed / lotteryEligibles.length) * lotteryEligibles.length));
679
680   }
681
682
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 678

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
677
678   return (seed - ((seed / lotteryEligibles.length) * lotteryEligibles.length));
679
680   }
681
682
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 678

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
677
678   return (seed - ((seed / lotteryEligibles.length) * lotteryEligibles.length));
679
680   }
681
682
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 678

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
677
678   return (seed - ((seed / lotteryEligibles.length) * lotteryEligibles.length));
679
680   }
681
682
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 682

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
681
682   function showWinner() external view returns(address)
683   {
684   // selectWinner();
685   return winner;
686
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
688
689   function swapAndLiquify(uint256 tokens) private lockTheSwap {
690
691   // split the contract balance into halves
692   uint256 half = tokens.div(2);
693
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 692

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
691   // split the contract balance into halves
692   uint256 half = tokens.div(2);
693   uint256 otherHalf = tokens.sub(half);
694
695   // capture the contract's current ETH balance.
696
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 692

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
691   // split the contract balance into halves
692   uint256 half = tokens.div(2);
693   uint256 otherHalf = tokens.sub(half);
694
695   // capture the contract's current ETH balance.
696
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 692

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
691   // split the contract balance into halves
692   uint256 half = tokens.div(2);
693   uint256 otherHalf = tokens.sub(half);
694
695   // capture the contract's current ETH balance.
696
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 438

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

- CRNOTOKEN.sol

Locations

```
437
438   bool inSwapAndLiquify;
439   bool public swapAndLiquifyEnabled = true;
440
441
442
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 423

low SEVERITY

The public state variable "lotteryEligibles" in "CRNOTOKEN" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- CRNOTOKEN.sol

Locations

```
422   uint256 public BUYmarketingFee = 2;
423   uint256 public BUYrewardFee = 2;
424   uint256 public BUYburnFee = 1;
425   uint256 public BUYliquidityFee = 1;
426   uint256 public BUYtotalFee =
BUYliquidityFee.add(BUYmarketingFee).add(BUYrewardFee).add(BUYburnFee);
427
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 674

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
673   block.gaslimit +
674   ((uint256(keccak256(abi.encodePacked(msg.sender)))) / (block.timestamp)) +
675   block.number
676   )));
677
678
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 743

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
742   path,
743   address(this),
744   block.timestamp
745   );
746   }
747
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 744

low SEVERITY

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

Source File

- CRNOTOKEN.sol

Locations

```
743   address(this),
744   block.timestamp
745   );
746   }
747
748
```

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

LINE 691

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

- CRNOTOKEN.sol

Locations

```
690
691   // split the contract balance into halves
692   uint256 half = tokens.div(2);
693   uint256 otherHalf = tokens.sub(half);
694
695
```

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.