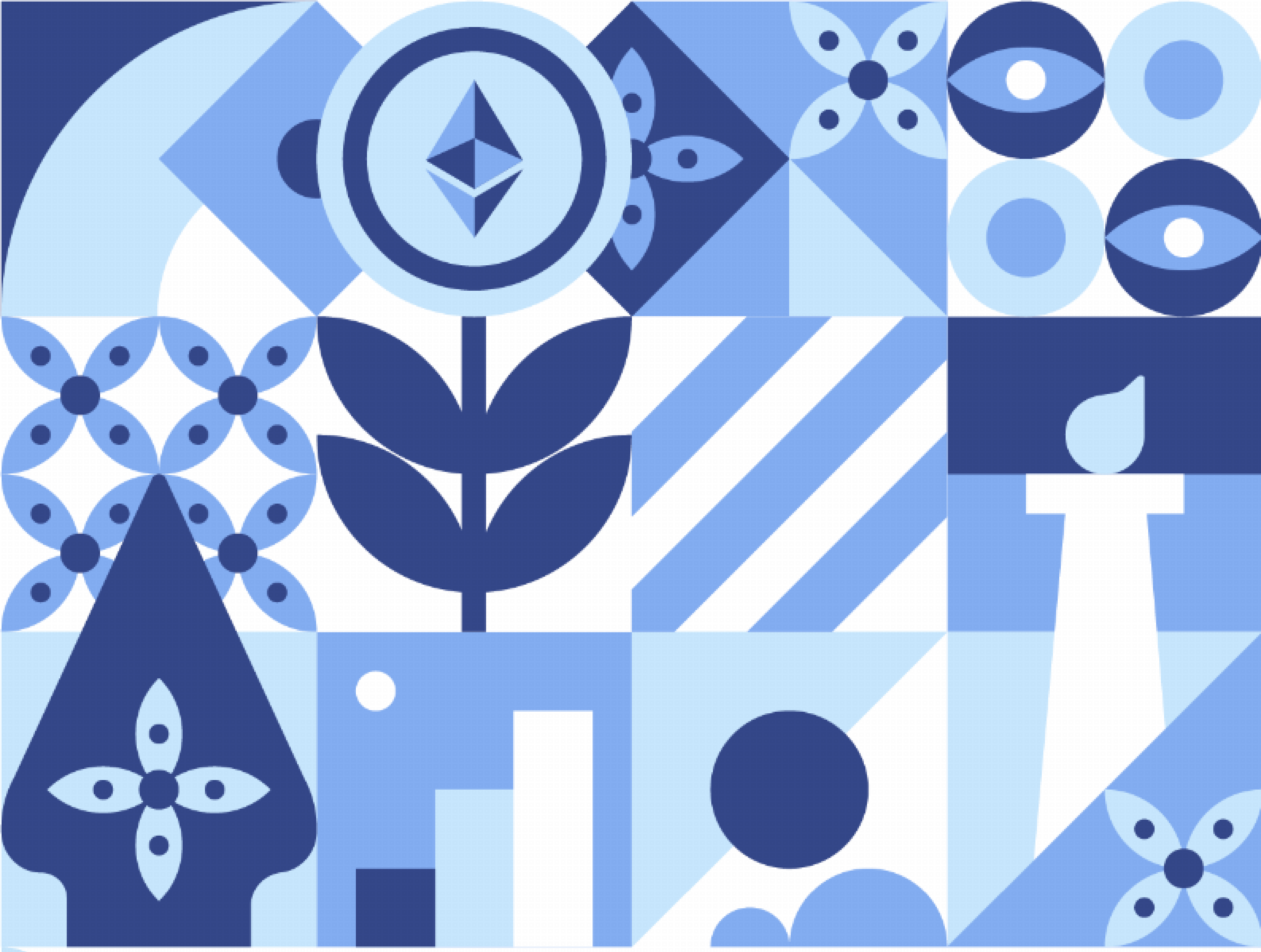

Chihiro Inu

Smart Contract Audit Report

SYSFIXED

31 Oct 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Chihiro Inu	CHIRO	Ethereum

Addresses

Contract address	0x35156b404c3f9bdaf45ab65ba315419bcde3775c
Contract deployer address	0x4dBe648F797E7eC51Da3eae23a89b971B4e022a5

Project Website

https://chihiro-inu.com/

Codebase

https://etherscan.io/address/0x35156b404c3f9bdaf45ab65ba315419bcde3775c#code

SUMMARY

Chihiro Inu is a next-generation GameFi ecosystem and launch pad founded in 2021. It is focusing on the development of high-quality play-to-earn games with excellent addictive gameplay and earning potential.

Contract Summary

Documentation Quality

Chihiro Inu 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 Chihiro Inu 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 699.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 95, 111, 117, 117, 133, 141, 660, 660, 661, 662, 694, 694, 701, 701, 706, 761, 761, 762, 762, 764, 764, 790, 790, 793, 793, 907, 919, 966, 970, 970, 977, 977, 1001, 1017, 1023, 1081, 1121, 1125, 1135, 1136, 1151, 1166, 1166, 1166, 1189, 1406, 1415, 1415, 1415, 1418, 1418, 1418, 1422, 1423, 1423, 1427, 1427, 1427, 1446, 1451, 1487, 1487, 1495, 1495 and 1023.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 43.
- 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 920, 921, 1020, 1022, 1022, 1197, 1197, 1407, 1408, 1409 and 1411.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 907, 1056, 1068 and 1071.

CONCLUSION

We have audited the Chihiro Inu project released on October 2021 to discover issues and identify potential security vulnerabilities in Chihiro Inu 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 Chihiro Inu smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
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	Thursday Oct 30 2021 07:39:53 GMT+0000 (Coordinated Universal Time)
Finished	Friday Oct 31 2021 20:25:14 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	ChihiroInu.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	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	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

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

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 95

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
94   uint256 c = a * b;
95   require(c / a == b, "SafeMath: multiplication overflow");
96
97   return c;
98   }
99
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 111

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
110   uint256 c = a / b;
111   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
112
113   return c;
114   }
115
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 117

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
116   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
117   return mod(a, b, "SafeMath: modulo by zero");
118   }
119
120   function mod(
121
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 117

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
116   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
117   return mod(a, b, "SafeMath: modulo by zero");
118   }
119
120   function mod(
121
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 133

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
132   // According to EIP-1052, 0x0 is the value returned for not-yet created accounts
133   // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is
returned
134   // for accounts without code, i.e. `keccak256('')`
135   bytes32 codehash;
136   bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
137
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 141

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
140   }
141   return (codehash != accountHash && codehash != 0x0);
142   }
143
144   function sendValue(address payable recipient, uint256 amount) internal {
145
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 660

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
659
660   uint256 private constant BUY = 1;
661   uint256 private constant SELL = 2;
662   uint256 private constant TRANSFER = 3;
663   uint256 private buyOrSellSwitch;
664
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 660

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
659
660   uint256 private constant BUY = 1;
661   uint256 private constant SELL = 2;
662   uint256 private constant TRANSFER = 3;
663   uint256 private buyOrSellSwitch;
664
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 661

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
660   uint256 private constant BUY = 1;
661   uint256 private constant SELL = 2;
662   uint256 private constant TRANSFER = 3;
663   uint256 private buyOrSellSwitch;
664
665
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 662

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
661   uint256 private constant SELL = 2;
662   uint256 private constant TRANSFER = 3;
663   uint256 private buyOrSellSwitch;
664
665   uint256 public _buyTaxFee = 2;
666
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 694

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
693
694   uint256 private minimumTokensBeforeSwap;
695
696   IUniswapV2Router02 public uniswapV2Router;
697   address public uniswapV2Pair;
698
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 694

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
693
694   uint256 private minimumTokensBeforeSwap;
695
696   IUniswapV2Router02 public uniswapV2Router;
697   address public uniswapV2Pair;
698
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 701

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
700   bool public swapAndLiquifyEnabled = false;
701   bool public tradingActive = false;
702
703   event SwapAndLiquifyEnabledUpdated(bool enabled);
704   event SwapAndLiquify(
705
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 701

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
700   bool public swapAndLiquifyEnabled = false;
701   bool public tradingActive = false;
702
703   event SwapAndLiquifyEnabledUpdated(bool enabled);
704   event SwapAndLiquify(
705
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 706

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
705   uint256 tokensSwapped,
706   uint256 ethReceived,
707   uint256 tokensIntoLiquidity
708   );
709
710
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 761

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
760   _isExcludedFromFee[address(this)] = true;
761   _isExcludedFromFee[marketingAddress] = true;
762   _isExcludedFromFee[liquidityAddress] = true;
763
764   excludeFromMaxTransaction(owner(), true);
765
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 761

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
760   _isExcludedFromFee[address(this)] = true;
761   _isExcludedFromFee[marketingAddress] = true;
762   _isExcludedFromFee[liquidityAddress] = true;
763
764   excludeFromMaxTransaction(owner(), true);
765
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 762

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
761   _isExcludedFromFee[marketingAddress] = true;
762   _isExcludedFromFee[liquidityAddress] = true;
763
764   excludeFromMaxTransaction(owner(), true);
765   excludeFromMaxTransaction(address(this), true);
766
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 762

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
761   _isExcludedFromFee[marketingAddress] = true;
762   _isExcludedFromFee[liquidityAddress] = true;
763
764   excludeFromMaxTransaction(owner(), true);
765   excludeFromMaxTransaction(address(this), true);
766
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 764

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
763
764   excludeFromMaxTransaction(owner(), true);
765   excludeFromMaxTransaction(address(this), true);
766   excludeFromMaxTransaction(address(0xdead), true);
767
768
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 764

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
763
764   excludeFromMaxTransaction(owner(), true);
765   excludeFromMaxTransaction(address(this), true);
766   excludeFromMaxTransaction(address(0xdead), true);
767
768
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 790

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
789   if (_isExcluded[account]) return _tOwned[account];
790   return tokenFromReflection(_rOwned[account]);
791   }
792
793   function transfer(address recipient, uint256 amount)
794
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 790

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
789   if (_isExcluded[account]) return _tOwned[account];
790   return tokenFromReflection(_rOwned[account]);
791   }
792
793   function transfer(address recipient, uint256 amount)
794
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 793

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```

function transfer(address recipient, uint256 amount)
external
override
returns (bool)

```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 793

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
792
793   function transfer(address recipient, uint256 amount)
794   external
795   override
796   returns (bool)
797
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 907

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
906   require(!tradingActive, "Trading is already active, cannot relaunch.");
907   require(presaleWallets.length < 200, "Can only airdrop 200 wallets per txn due to
gas limits"); // allows for airdrop + launch at the same exact time, reducing delays and
reducing sniper input.
908   for(uint256 i = 0; i < presaleWallets.length; i++){
909   address wallet = presaleWallets[i];
910   uint256 amount = amounts[i];
911
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 919

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
918   uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
919   excludeFromMaxTransaction(address(uniswapV2Pair), true);
920   _setAutomatedMarketMakerPair(address(uniswapV2Pair), true);
921   require(address(this).balance > 0, "Must have ETH on contract to launch");
922   addLiquidity(balanceOf(address(this)), address(this).balance);
923
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 966

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
965   return rAmount;
966   } else {
967   (, uint256 rTransferAmount, , , , ) = _getValues(tAmount);
968   return rTransferAmount;
969   }
970
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 970

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
969   }
970   }
971
972   function tokenFromReflection(uint256 rAmount)
973   public
974
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 970

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
969   }
970   }
971
972   function tokenFromReflection(uint256 rAmount)
973   public
974
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
976   {
977   require(
978   rAmount <= _rTotal,
979   "Amount must be less than total reflections"
980   );
981
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
976   {
977   require(
978   rAmount <= _rTotal,
979   "Amount must be less than total reflections"
980   );
981
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1001

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1000   _tOwned[account] = 0;
1001   _isExcluded[account] = false;
1002   _excluded.pop();
1003   break;
1004   }
1005
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1017

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1016   _allowances[owner][spender] = amount;
1017   emit Approval(owner, spender, amount);
1018   }
1019
1020   function _transfer(
1021
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1023

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1022   address to,
1023   uint256 amount
1024   ) private {
1025   require(from != address(0), "ERC20: transfer from the zero address");
1026   require(to != address(0), "ERC20: transfer to the zero address");
1027
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1081

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1080   uint256 contractTokenBalance = balanceOf(address(this));
1081   bool overMinimumTokenBalance = contractTokenBalance >= minimumTokensBeforeSwap;
1082
1083   // swap and liquify
1084   if (
1085
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1121

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1120   _liquidityFee = _liquidityFee * 10;
1121   }
1122   // Normal transfers do not get taxed
1123   } else {
1124   require(!boughtEarly[from] || earlyBuyPenaltyEnd <= block.timestamp, "Snipers
can't transfer tokens to sell cheaper until penalty timeframe is over.  DM a Mod.");
1125
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1125

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1124   require(!boughtEarly[from] || earlyBuyPenaltyEnd <= block.timestamp, "Snipers
can't transfer tokens to sell cheaper until penalty timeframe is over.  DM a Mod.");
1125   removeAllFee();
1126   buyOrSellSwitch = TRANSFER; // TRANSFERs do not pay a tax.
1127   }
1128   }
1129
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1135

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1134   function swapBack() private lockTheSwap {
1135   uint256 contractBalance = balanceOf(address(this));
1136   uint256 totalTokensToSwap = _liquidityTokensToSwap + _marketingTokensToSwap;
1137
1138   // Halve the amount of liquidity tokens
1139
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1136

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1135   uint256 contractBalance = balanceOf(address(this));
1136   uint256 totalTokensToSwap = _liquidityTokensToSwap + _marketingTokensToSwap;
1137
1138   // Halve the amount of liquidity tokens
1139   uint256 tokensForLiquidity = _liquidityTokensToSwap.div(2);
1140
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1151

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1150   uint256 ethForLiquidity = ethBalance.sub(ethForMarketing);
1151
1152   uint256 ethForDev= ethForMarketing * 2 / 7; // 2/7 gos to dev
1153   ethForMarketing -= ethForDev;
1154
1155
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1166

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1165   if(address(this).balance > 1e17){
1166   (success,) = address(marketingAddress).call{value: address(this).balance}("");
1167   }
1168   }
1169
1170
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1166

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1165   if(address(this).balance > 1e17){
1166   (success,) = address(marketingAddress).call{value: address(this).balance}("");
1167   }
1168   }
1169
1170
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1166

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1165   if(address(this).balance > 1e17){
1166   (success,) = address(marketingAddress).call{value: address(this).balance}("");
1167   }
1168   }
1169
1170
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1189

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1188   block.timestamp
1189   );
1190   }
1191
1192   function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
1193
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1406

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1405   _liquidityTokensToSwap += tLiquidity * _buyLiquidityFee / _liquidityFee;
1406   _marketingTokensToSwap += tLiquidity * _buyMarketingFee / _liquidityFee;
1407   } else if(buyOrSellSwitch == SELL){
1408   _liquidityTokensToSwap += tLiquidity * _sellLiquidityFee / _liquidityFee;
1409   _marketingTokensToSwap += tLiquidity * _sellMarketingFee / _liquidityFee;
1410
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1415

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1414   if (_isExcluded[address(this)])
1415   _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity);
1416   }
1417
1418   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1419
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1415

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1414   if (_isExcluded[address(this)])
1415   _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity);
1416   }
1417
1418   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1419
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1415

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1414   if (_isExcluded[address(this)])
1415   _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity);
1416   }
1417
1418   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1419
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1418

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1417
1418   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1419   return _amount.mul(_taxFee).div(10**2);
1420   }
1421
1422
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1418

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1417
1418   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1419   return _amount.mul(_taxFee).div(10**2);
1420   }
1421
1422
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1418

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```

function calculateTaxFee(uint256 _amount) private view returns (uint256) {
return _amount.mul(_taxFee).div(10**2);
}


```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1422

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1421
1422   function calculateLiquidityFee(uint256 _amount)
1423   private
1424   view
1425   returns (uint256)
1426
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1423

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1422    function calculateLiquidityFee(uint256 _amount)
1423    private
1424    view
1425    returns (uint256)
1426    {
1427
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1423

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1422   function calculateLiquidityFee(uint256 _amount)
1423   private
1424   view
1425   returns (uint256)
1426   {
1427
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1427

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1426   {
1427   return _amount.mul(_liquidityFee).div(10**2);
1428   }
1429
1430   function removeAllFee() private {
1431
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1427

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1426   {
1427   return _amount.mul(_liquidityFee).div(10**2);
1428   }
1429
1430   function removeAllFee() private {
1431
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1427

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1426   {
1427   return _amount.mul(_liquidityFee).div(10**2);
1428   }
1429
1430   function removeAllFee() private {
1431
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1446

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1445   function isExcludedFromFee(address account) external view returns (bool) {
1446   return _isExcludedFromFee[account];
1447   }
1448
1449   function removeBoughtEarly(address account) external onlyOwner {
1450
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1451

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1450   boughtEarly[account] = false;
1451   emit RemovedSniper(account);
1452   }
1453
1454   function excludeFromFee(address account) external onlyOwner {
1455
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1487

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1486
1487   function setMarketingAddress(address _marketingAddress) external onlyOwner {
1488   require(_marketingAddress != address(0), "_marketingAddress address cannot be 0");
1489   _isExcludedFromFee[marketingAddress] = false;
1490   marketingAddress = payable(_marketingAddress);
1491
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1487

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1486
1487   function setMarketingAddress(address _marketingAddress) external onlyOwner {
1488   require(_marketingAddress != address(0), "_marketingAddress address cannot be 0");
1489   _isExcludedFromFee[marketingAddress] = false;
1490   marketingAddress = payable(_marketingAddress);
1491
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1495

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1494
1495   function setLiquidityAddress(address _liquidityAddress) public onlyOwner {
1496   require(_liquidityAddress != address(0), "_liquidityAddress address cannot be 0");
1497   liquidityAddress = payable(_liquidityAddress);
1498   _isExcludedFromFee[liquidityAddress] = true;
1499
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1495

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1494
1495   function setLiquidityAddress(address _liquidityAddress) public onlyOwner {
1496   require(_liquidityAddress != address(0), "_liquidityAddress address cannot be 0");
1497   liquidityAddress = payable(_liquidityAddress);
1498   _isExcludedFromFee[liquidityAddress] = true;
1499
```

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

LINE 1023

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1022   address to,
1023   uint256 amount
1024   ) private {
1025   require(from != address(0), "ERC20: transfer from the zero address");
1026   require(to != address(0), "ERC20: transfer to the zero address");
1027
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 43

low SEVERITY

The current pragma Solidity directive is ""^0.8.9"". 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

- ChihiroInu.sol

Locations

```
42   external
43   returns (bool);
44
45   function allowance(address owner, address spender)
46   external
47
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 699

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

- ChihiroInu.sol

Locations

```
698
699   bool inSwapAndLiquify;
700   bool public swapAndLiquifyEnabled = false;
701   bool public tradingActive = false;
702
703
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 920

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
919   excludeFromMaxTransaction(address(uniswapV2Pair), true);
920   _setAutomatedMarketMakerPair(address(uniswapV2Pair), true);
921   require(address(this).balance > 0, "Must have ETH on contract to launch");
922   addLiquidity(balanceOf(address(this)), address(this).balance);
923   setLiquidityAddress(address(0xdead));
924
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 921

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
920   _setAutomatedMarketMakerPair(address(uniswapV2Pair), true);
921   require(address(this).balance > 0, "Must have ETH on contract to launch");
922   addLiquidity(balanceOf(address(this)), address(this).balance);
923   setLiquidityAddress(address(0xdead));
924   return true;
925
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1020

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1019
1020   function _transfer(
1021   address from,
1022   address to,
1023   uint256 amount
1024
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1022

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1021   address from,
1022   address to,
1023   uint256 amount
1024   ) private {
1025   require(from != address(0), "ERC20: transfer from the zero address");
1026
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1022

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1021   address from,
1022   address to,
1023   uint256 amount
1024   ) private {
1025   require(from != address(0), "ERC20: transfer from the zero address");
1026
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1197

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1196   tokenAmount,
1197   0, // slippage is unavoidable
1198   0, // slippage is unavoidable
1199   liquidityAddress,
1200   block.timestamp
1201
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1197

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1196   tokenAmount,
1197   0, // slippage is unavoidable
1198   0, // slippage is unavoidable
1199   liquidityAddress,
1200   block.timestamp
1201
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1407

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1406   _marketingTokensToSwap += tLiquidity * _buyMarketingFee / _liquidityFee;
1407   } else if(buyOrSellSwitch == SELL){
1408   _liquidityTokensToSwap += tLiquidity * _sellLiquidityFee / _liquidityFee;
1409   _marketingTokensToSwap += tLiquidity * _sellMarketingFee / _liquidityFee;
1410   }
1411
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1408

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1407   } else if(buyOrSellSwitch == SELL){
1408   _liquidityTokensToSwap += tLiquidity * _sellLiquidityFee / _liquidityFee;
1409   _marketingTokensToSwap += tLiquidity * _sellMarketingFee / _liquidityFee;
1410   }
1411   uint256 currentRate = _getRate();
1412
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1409

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1408   _liquidityTokensToSwap += tLiquidity * _sellLiquidityFee / _liquidityFee;
1409   _marketingTokensToSwap += tLiquidity * _sellMarketingFee / _liquidityFee;
1410   }
1411   uint256 currentRate = _getRate();
1412   uint256 rLiquidity = tLiquidity.mul(currentRate);
1413
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1411

low SEVERITY

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

Source File

- ChihiroInu.sol

Locations

```
1410   }
1411   uint256 currentRate = _getRate();
1412   uint256 rLiquidity = tLiquidity.mul(currentRate);
1413   _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity);
1414   if (_isExcluded[address(this)])
1415
```

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

LINE 907

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

- ChihiroInu.sol

Locations

```
906   require(!tradingActive, "Trading is already active, cannot relaunch.");
907   require(presaleWallets.length < 200, "Can only airdrop 200 wallets per txn due to
gas limits"); // allows for airdrop + launch at the same exact time, reducing delays and
reducing sniper input.
908   for(uint256 i = 0; i < presaleWallets.length; i++){
909   address wallet = presaleWallets[i];
910   uint256 amount = amounts[i];
911
```

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

LINE 1056

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

- ChihiroInu.sol

Locations

```
1055   if (transferDelayEnabled){
1056   if (to != owner() && to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1057   require(_holderLastTransferTimestamp[to] < block.number, "_transfer:: Transfer Delay enabled.  Only one purchase per block allowed.");
1058   _holderLastTransferTimestamp[to] = block.number;
1059   }
1060
```

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

LINE 1068

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

- ChihiroInu.sol

Locations

```
1067   if (to != address(uniswapV2Router) && !automatedMarketMakerPairs[to]) {
1068   require(balanceOf(to) + amount <= maxWalletSize, "Exceeds the maxWalletSize.");
1069   }
1070   //when sell
1071   else if (automatedMarketMakerPairs[to] && !_isExcludedMaxTransactionAmount[from])
{
1072
```

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

LINE 1071

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

- ChihiroInu.sol

Locations

```
1070   //when sell
1071   else if (automatedMarketMakerPairs[to] && !_isExcludedMaxTransactionAmount[from])
{
1072   require(amount <= maxTransactionAmount, "Sell transfer amount exceeds the
maxTransactionAmount.");
1073   }
1074   }
1075
```

DISCLAIMER

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.