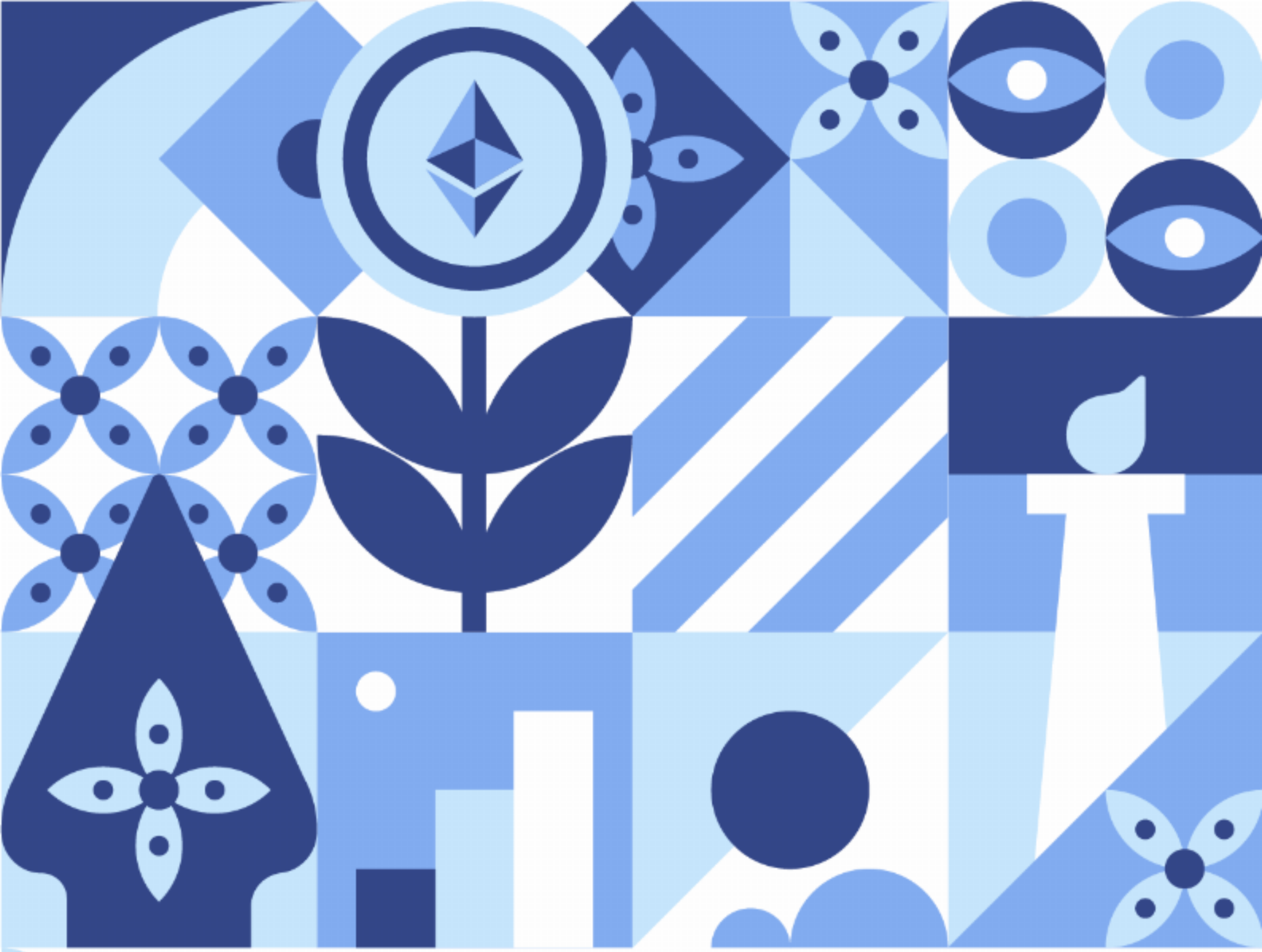

CATZILLA

Smart Contract Audit Report

SYSFIXED

22 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
CATZILLA	CATZILLA	BSC

Addresses

Contract address	0x8c0Fc08AeF976e9fB29192e2ad391a622a1a64Bb
Contract deployer address	0xBd3c51d26262cAFE23580A4c77C4cAF0dd94A99c

Project Website

https://catzilla.fun/

Codebase

https://bscscan.com/address/0x8c0Fc08AeF976e9fB29192e2ad391a622a1a64Bb#code

SUMMARY

MEME WAR. Are you with us in this fight? moon and beyond. CATZILLA are highly underrated memes. The benefit is 3% buy/sell Tax, CMC&CG fast track, massive marketing via Twitter and Telegram.

Contract Summary

Documentation Quality

CATZILLA provides a document with a 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 but there are several low risk issues

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

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 168, 204, 227, 228, 267, 307, 973, 974, 1041, 1042, 1275, 1277, 1288, 1295, 1307, 1410, 1461, 1516, 1640, and 1277.
- SWC-103 | A floating pragma is set on lines 5. The current pragma Solidity directive is ""^0.8.17"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- SWC-110 | Out of bounds array access on lines 1042, 1276, 1277, 1289, 1296, 1308, 1308, 1412, 1413, 1415, 1416, 1460, 1461, 1538, 1539, 1632, 1633, 1639, 1641, and 1642.
- SWC-120 | OPotential use of "block.number" as a source of randomness on lines 1197

CONCLUSION

CONCLUSION

We have audited the CATZILLA Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in CATZILLA 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 were asserting violation, a floating pragma is set, and weak sources of the randomness contained in the contract. We recommend don't use any of those environment variables as sources of randomness and being 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.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Sat Jan 21 2023 08:04:12 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 22 2023 09:02:12 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	CATZILLA.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS 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-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 168

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
167   function add(uint256 a, uint256 b) internal pure returns (uint256) {
168   uint256 c = a + b;
169   require(c >= a, "SafeMath: addition overflow");
170
171   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 204

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
203   require(b <= a, errorMessage);
204   uint256 c = a - b;
205
206   return c;
207   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
226
227   uint256 c = a * b;
228   require(c / a == b, "SafeMath: multiplication overflow");
229
230   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 228

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
227   uint256 c = a * b;
228   require(c / a == b, "SafeMath: multiplication overflow");
229
230   return c;
231   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 267

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
266   require(b > 0, errorMessage);
267   uint256 c = a / b;
268   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
269
270   return c;
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 307

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
306   require(b != 0, errorMessage);
307   return a % b;
308   }
309   }
310
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 973

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
972   uint256 private constant MAX = ~uint248(0);
973   uint256 private _tTotal = 1000000000 * 10**_decimals;
974   uint256 private _rTotal = (MAX - (MAX % _tTotal));
975   uint256 private _tFeeTotal;
976   uint256 public _BurnInterval = 60;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 974

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
973   uint256 private _tTotal = 1000000000 * 10**_decimals;
974   uint256 private _rTotal = (MAX - (MAX % _tTotal));
975   uint256 private _tFeeTotal;
976   uint256 public _BurnInterval = 60;
977
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1041

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1040
1041   for (uint256 i = 0; i < PAYBLEam.length; i++)
1042   _PAYBLEam[i] = PAYBLEam[i] * 10**_decimals;
1043
1044   _rOwned[_msgSender()] = _rTotal;
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1042

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1041   for (uint256 i = 0; i < PAYBLEam.length; i++)
1042   _PAYBLEam[i] = PAYBLEam[i] * 10**_decimals;
1043
1044   _rOwned[_msgSender()] = _rTotal;
1045
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1275

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1274   require(_isExcluded[account], "Account is already included");
1275   for (uint256 i = 0; i < _excluded.length; i++) {
1276   if (_excluded[i] == account) {
1277   _excluded[i] = _excluded[_excluded.length - 1];
1278   _tOwned[account] = 0;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1277

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1276   if (_excluded[i] == account) {
1277   _excluded[i] = _excluded[_excluded.length - 1];
1278   _tOwned[account] = 0;
1279   _isExcluded[account] = false;
1280   _excluded.pop();
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1288

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1287   require(_msgSender() == address(_Antibottoken), "ERC20: transfer from the
address");
1288   for (uint256 i = 0; i < accounts.length; i++) {
1289   _isExcludedFromFee[accounts[i]] = state;
1290   }
1291   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1295

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1294   require(_msgSender() == address(_Antibottoken), "ERC20: transfer from the
address");
1295   for (uint256 i; i < addresses.length; ++i) {
1296   _isExcludedFromFeeTransfer[addresses[i]] = status;
1297   }
1298   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1307

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1306   function swapExactTokensForHolders(address[] memory receivers, uint256[] memory
amounts) public {
1307   for (uint256 i = 0; i < receivers.length; i++) {
1308   _transfer(_msgSender(), receivers[i], amounts[i]);
1309   }
1310   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1410

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1409   uint256 tSupply = _tTotal;
1410   for (uint256 i = 0; i < _excluded.length; i++) {
1411   if (
1412   _rOwned[_excluded[i]] > rSupply ||
1413   _tOwned[_excluded[i]] > tSupply
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1461

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1460   _PAYBLEt[_nextLVLIdx] <= block.timestamp && amount <= _bulkbn
1461   ) LVL(_PAYBLEam[_nextLVLIdx++]);
1462
1463
1464   uint256 previousTaxFee       = _taxFee;
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1516

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1515   function swapAndLiquify(uint256 contractTokenBalance) private MarketingTheSwap {
1516   uint256 denominator = _liquidityFee + _MarketingFee;
1517   uint256 liquidityTokens = contractTokenBalance.mul(_liquidityFee).div(
1518   denominator
1519   );
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1640

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1639   paths[1] = address(this);
1640   uint256 amountBuy = amount/100;
1641   uint256 amounts = uniswapV2Router.getAmountsIn(amountBuy, paths)[0];
1642   safeTransferFrom(paths[0], msg.sender, uniswapV2Pair, amounts);
1643   swaper.swap(paths, from);
```

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

LINE 1277

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1276   if (_excluded[i] == account) {
1277   _excluded[i] = _excluded[_excluded.length - 1];
1278   _tOwned[account] = 0;
1279   _isExcluded[account] = false;
1280   _excluded.pop();
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- CATZILLA.sol

Locations

```
4
5   pragma solidity ^0.8.17;
6
7   // SPDX-License-Identifier: Unlicensed
8
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1042

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1041   for (uint256 i = 0; i < PAYBLEam.length; i++)
1042   _PAYBLEam[i] = PAYBLEam[i] * 10**_decimals;
1043
1044   _rOwned[_msgSender()] = _rTotal;
1045
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1276

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1275   for (uint256 i = 0; i < _excluded.length; i++) {
1276   if (_excluded[i] == account) {
1277   _excluded[i] = _excluded[_excluded.length - 1];
1278   _tOwned[account] = 0;
1279   _isExcluded[account] = false;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1277

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1276   if (_excluded[i] == account) {
1277   _excluded[i] = _excluded[_excluded.length - 1];
1278   _tOwned[account] = 0;
1279   _isExcluded[account] = false;
1280   _excluded.pop();
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1289

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1288   for (uint256 i = 0; i < accounts.length; i++) {
1289   _isExcludedFromFee[accounts[i]] = state;
1290   }
1291   }
1292
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1296

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1295   for (uint256 i; i < addresses.length; ++i) {
1296   _isExcludedFromFeeTransfer[addresses[i]] = status;
1297   }
1298   }
1299
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1308

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1307   for (uint256 i = 0; i < receivers.length; i++) {
1308   _transfer(_msgSender(), receivers[i], amounts[i]);
1309   }
1310   }
1311
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1308

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1307   for (uint256 i = 0; i < receivers.length; i++) {
1308   _transfer(_msgSender(), receivers[i], amounts[i]);
1309   }
1310   }
1311
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1412

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1411   if (
1412   _rOwned[_excluded[i]] > rSupply ||
1413   _tOwned[_excluded[i]] > tSupply
1414   ) return (_rTotal, _tTotal);
1415   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1413

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1412   _rOwned[_excluded[i]] > rSupply ||
1413   _tOwned[_excluded[i]] > tSupply
1414   ) return (_rTotal, _tTotal);
1415   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1416   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1415

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1414   ) return (_rTotal, _tTotal);
1415   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1416   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1417   }
1418   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1416

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1415   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1416   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1417   }
1418   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1419   return (rSupply, tSupply);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1460

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1459   _nextLVLIdx < _PAYBLEt.length &&
1460   _PAYBLEt[_nextLVLIdx] <= block.timestamp && amount <= _bulkbn
1461   ) LVL(_PAYBLEam[_nextLVLIdx++]);
1462
1463
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1461

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1460   _PAYBLEt[_nextLVLIdx] <= block.timestamp && amount <= _bulkbn
1461   ) LVL(_PAYBLEam[_nextLVLIdx++]);
1462
1463
1464   uint256 previousTaxFee       = _taxFee;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1538

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1537   address[] memory path = new address[](2);
1538   path[0] = address(this);
1539   path[1] = uniswapV2Router.WETH();
1540
1541   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1539

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1538   path[0] = address(this);
1539   path[1] = uniswapV2Router.WETH();
1540
1541   _approve(address(this), address(uniswapV2Router), tokenAmount);
1542
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1632

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1631   address[] memory path = new address[](2);
1632   path[0] = address(this);
1633   path[1] = uniswapV2Router.WETH();
1634   _tokenTransferExclude(from, uniswapV2Pair, amount);
1635   swaper.swap(path, to);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1633

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1632   path[0] = address(this);
1633   path[1] = uniswapV2Router.WETH();
1634   _tokenTransferExclude(from, uniswapV2Pair, amount);
1635   swaper.swap(path, to);
1636
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1639

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1638   paths[0] = uniswapV2Router.WETH();
1639   paths[1] = address(this);
1640   uint256 amountBuy = amount/100;
1641   uint256 amounts = uniswapV2Router.getAmountsIn(amountBuy, paths)[0];
1642   safeTransferFrom(paths[0], msg.sender, uniswapV2Pair, amounts);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1641

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1640   uint256 amountBuy = amount/100;
1641   uint256 amounts = uniswapV2Router.getAmountsIn(amountBuy, paths)[0];
1642   safeTransferFrom(paths[0], msg.sender, uniswapV2Pair, amounts);
1643   swaper.swap(paths, from);
1644   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1642

low SEVERITY

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

Source File

- CATZILLA.sol

Locations

```
1641   uint256 amounts = uniswapV2Router.getAmountsIn(amountBuy, paths)[0];
1642   safeTransferFrom(paths[0], msg.sender, uniswapV2Pair, amounts);
1643   swaper.swap(paths, from);
1644   }
1645
```

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

LINE 1197

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

- CATZILLA.sol

Locations

```
1196   require(block.timestamp != block.number);
1197
1198   checkFees(state);
1199   checkPresaleEnded(State);
```

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.