Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	CateFarm Token
Website:	https://catefarm.io
Platform:	Binance Smart Chain
Language:	Solidity
Date:	April 8th, 2022

Table of contents

Email: audit@EtherAuthority.io

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the CateFarm team to perform the Security audit of the CateFarm Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 8th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

CateFarm is a standard BEP20 token smart contract. This audit only considers the CateFarm token smart contract, and does not cover any other smart contracts on the platform.

Audit scope

Name	**Code Review and Security Analysis Report for CateFarm Token Smart Contract**
Platform	**BSC / Solidity**
File	CATEFARM.sol
File MD5 Hash	5FECB6D6D4CC19978765F99E57F4B5E6
Updated File MD5 Hash	DB5D499ED0642DBA171C885299ED2A76
Online Code Link	0x5de624fcfd372E34cE0ed6d63519feC04c791c56
Audit Date	April 8th, 2022
Revise Audit Date	April 14th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: CateFarm • Symbol: CATEFARM • Decimals: 18 • Total Supply: 1 Billion • Swap Threshold: 0.5 Million • Swap Amount: 1 Million • Minimum Tokens for rewards: 50,000 • Maximum Transfer Taxes: 15% • Maximum Buy Taxes: 15% • Maximum Sell Taxes: 15% • Get Maximum Wallet: 1 Billion • Get Maximum transfer: 1 Billion • Reflector Gas: 0.75 Million	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
• **Ratios:** ○ Rewards: 16% ○ Liquidity : 6% ○ Marketing : 2% ○ Team: 2% ○ Total : 26%	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
• **Tax Rates:** ○ buyFee: 13% ○ sellFee : 13% ○ transfer: 0%	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 2 low and some very low level issues.
All these issues have been resolved / acknowledged.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract file. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in CateFarm Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the CateFarm Token.

The CateFarm Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not well** commented on smart contracts.

Documentation

We were given a CateFarm Token smart contract code in the form of a BSCScan Web Link.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website http://catefarm.io/ which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	swapping	modifier	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	transferOwner	external	access only Owner	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	receive	external	Passed	No Issue
7	totalSupply	external	Passed	No Issue
8	decimals	external	Passed	No Issue
9	symbol	external	Passed	No Issue
10	getOwner	external	Passed	No Issue
11	name	external	Passed	No Issue
12	balanceOf	read	Passed	No Issue
13	allowance	external	Passed	No Issue
14	approve	write	Passed	No Issue
15	_approve	write	Passed	No Issue
16	approveContractContingency	write	access only Owner	No Issue
17	transfer	external	Passed	No Issue
18	transferFrom	external	Passed	No Issue
19	setBlacklistEnabled	external	access only Owner	No Issue
20	setBlacklistEnabledMultiple	external	access only Owner	No Issue
21	isBlacklisted	read	Passed	No Issue
22	setInitializers	external	access only Owner	No Issue
23	removeSniper	external	access only Owner	No Issue
24	setProtectionSettings	external	access only Owner	No Issue
25	setGasPriceLimit	external	access only Owner	No Issue
26	enableTrading	write	access only Owner	No Issue
27	setTaxes	external	access only Owner	No Issue
28	setRatios	external	access only Owner	No Issue
29	setWallets	external	access only Owner	No Issue
30	setContractSwapSettings	external	access only Owner	No Issue
31	setSwapSettings	external	access only Owner	No Issue
32	setReflectionCriteria	external	access only Owner	No Issue
33	setReflectorSettings	external	access only Owner	No Issue
34	claimRewards	external	Critical operation lacks event log	Refer Audit Findings
35	getTotalReflected	external	Passed	No Issue
36	getUserInfo	external	Passed	No Issue
37	getUserRealizedGains	external	Passed	No Issue
38	getUserUnpaidEarnings	external	Passed	No Issue
39	setNewRouter	write	access only Owner	No Issue
40	setLpPair	external	access only Owner	No Issue
41	isExcludedFromFees	read	Passed	No Issue

42	isExcludedFromDividends	read	Passed	No Issue
43	isExcludedFromLimits	read	Passed	No Issue
44	setExcludedFromLimits	external	access only Owner	No Issue
45	setDividendExcluded	write	access only Owner	No Issue
46	setExcludedFromFees	write	access only Owner	No Issue
47	setMaxTxPercent	external	access only Owner	No Issue
48	setMaxWalletSize	external	access only Owner	No Issue
49	getMaxTX	read	Passed	No Issue
50	getMaxWallet	read	Passed	No Issue
51	excludePresaleAddresses	external	access only Owner	No Issue
52	_hasLimits	write	Passed	No Issue
53	_transfer	internal	Passed	No Issue
54	_finalizeTransfer	internal	Passed	No Issue
55	processTokenReflect	internal	Passed	No Issue
56	_basicTransfer	internal	Passed	No Issue
57	takeTaxes	Passed	No Issue	Passed
58	contractSwap	internal	Passed	No Issue
59	_checkLiquidityAdd	write	Passed	No Issue
60	multiSendTokens	external	Infinite loops possibility	Refer Audit Findings
61	manualDeposit	external	access only Owner	No Issue
62	setMinimumTokensForRewards	external	access only Owner	No Issue
63	whomst_tokens	external	Passed	No Issue
64	whomst_routers	external	Passed	No Issue
65	updateRewardsTokens	external	Passed	No Issue
66	getRewardsRatios	external	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:
Missing event log for: claimRewards

Resolution: Please write an event log for listed events.
Status: Acknowledged

(2) Infinite loops possibility:

```
function multiSendTokens(address[] memory accounts, uint256[] memory amounts) external onlyOwner {
    require(accounts.length == amounts.length, "Lengths do not match.");
    for (uint8 i = 0; i < accounts.length; i++) {
        require(balanceOf(msg.sender) >= amounts[i]);
        _finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false, false);
    }
}
```

As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.

Resolution: Adjust logic to replace loops with mapping or other code structure.

- multiSendTokens() - accounts.length

Status: Acknowledged

Very Low / Informational / Best practices:

(1) Missing required error message:

```
function setDividendExcluded(address holder, bool enabled) public onlyOwner {
    require(holder != address(this) && holder != lpPair);
```

```
function setReflectorSettings(uint256 gas) external onlyOwner {
    require(gas < 750000);
    reflectorGas = gas;
}
```

```
function excludePresaleAddresses(address router, address presale) external onlyOwner {
    require(allowedPresaleExclusion);
    if (router == presale) {
```

```
function setTaxes(uint16 buyFee, uint16 sellFee, uint16 transferFee) external onlyOwner {
    require(buyFee <= maxBuyTaxes
            && sellFee <= maxSellTaxes
            && transferFee <= maxTransferTaxes);
    _taxRates.buyFee = buyFee;
    _taxRates.sellFee = sellFee;
```

```
function setInitializers(address aInitializer, address cInitializer) external onlyOwner {
    require(!_hasLiqBeenAdded);
    require(cInitializer != address(this) && aInitializer != address(this) && cInitializer != aInitializer);
```

There is no error message set in the required condition.

Resolution: We suggest setting relevant error messages to identify the failure of the transaction.

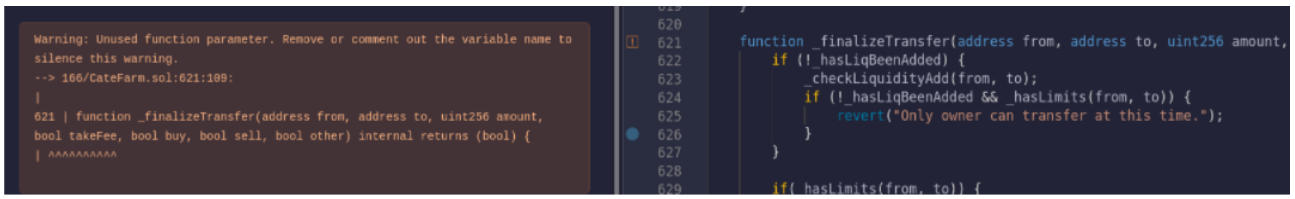
Status: Fixed

(2) Unused function parameter / variables / event / internal function:

Unused function parameter:

```
Warning: Unused function parameter. Remove or comment out the
variable name to silence this warning.
--> 166/CateFarm.sol:683:75:
|
683 | function takeTaxes(address from, uint256 amount, bool
buy, bool sell, bool other) internal returns (uint256) {
| ^^^^^^^^^^
```

```
681     }
682
683     function takeTaxes(address from, uint256 amount, bool buy, bool sell, bool other)
684         uint256 currentFee;
685         if (buy) {
686             currentFee = _taxRates.buyFee;
687         } else if (sell) {
688             currentFee = _taxRates.sellFee;
ContractDefinition TESTINGTIME  1 reference(s)  ransferFee;
```



Unused variables:

```
address constant public CATECOIN = 0xE4FAE3Faa8300810C835970b9187c268f55D998F;
address constant public CATPAY = 0x0611686A2558de495617685b3Da12448657170FE;
```

```
function _transfer(address from, address to, uint256 amount) internal returns (bool) {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    bool buy = false;
    bool sell = false;
    bool other = false;
    if (lpPairs[from]) {
        buy = true;
    } else if (lpPairs[to]) {
        sell = true;
    } else {
        other = true;
    }
    if(_hasLimits(from, to)) {
        if(!tradingEnabled) {
```

Unused event:

```
event SniperCaught(address sniperAddress);
```

Unused internal function:

```
function _basicTransfer(address from, address to, uint256 amount) internal returns (bool) {
    _tOwned[from] -= amount;
    _tOwned[to] += amount;
    emit Transfer(from, to, amount);
    return true;
}
```

There are many functions that have passed unused function parameters. CATECOIN, CATPAY, and other variables defined but not used anywhere.A SniperCaught() event is defined but not used in code. A _basicTransfer internal function defined but not used anywhere.

Resolution: Remove unused variables / event / function parameter / internal function from the code.

Status: Fixed

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- transferOwner: Owner can be removed as a library and added here to allow for custom transfers and announcements.
- renounceOwnership: Owner can renounce new ownership.
- approveContractContingency: Owner can approve contract contingency.
- setBlacklistEnabled: Owner can set enabled status in address in blacklist.
- setInitializers: Owner can set initializers.
- removeSniper: Owner can remove sniper address.
- setProtectionSettings: Owner can set protection settings like: _antiSnipe, _antiGas, _antiBlock, _algo.
- setGasPriceLimit: Owner can set gas price limit.
- enableTrading: Owner can enable trading status.
- setTaxes: Owner can set buy Fee, sell Fee, transfer Fee taxes.
- setRatios: Owner can set rewardsToken1, rewardsToken2, liquidity, marketing ,team ratios.
- setWallets: Owner can set wallet addresses like: marketing address, payable team address, liquidity address.
- setContractSwapSettings: Owner can set contract swap settings status.
- setSwapSettings: Owner can set swap settings like: thresholdPercent, thresholdPercent, amountPercent, amountDivisor.
- setReflectionCriteria: Owner can set reflection criteria like: _minPeriod, _minReflection, minReflectionMultiplier.
- setReflectorSettings: Owner can set reflector value.
- setNewRouter: Owner can set new router address.
- setLpPair: Owner can set LP pair address.
- setExcludedFromLimits: Owner can set excluded address status.
- setDividendExcluded: Owner can set dividend exclude address.
- setExcludedFromFees: Owner can set excluded fees from account.

- setMaxTxPercent: Owner can set transaction percentage.
- setMaxWalletSize: Owner can set maximum wallet size.
- excludePresaleAddresses: Owner can set presale address and router address.
- multiSendTokens: Owner can send multi tokens.
- manualDeposit: Owner can manual deposit.
- setMinimumTokensForRewards: Owner can set minimum tokens for rewards.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We have not observed any major issues. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

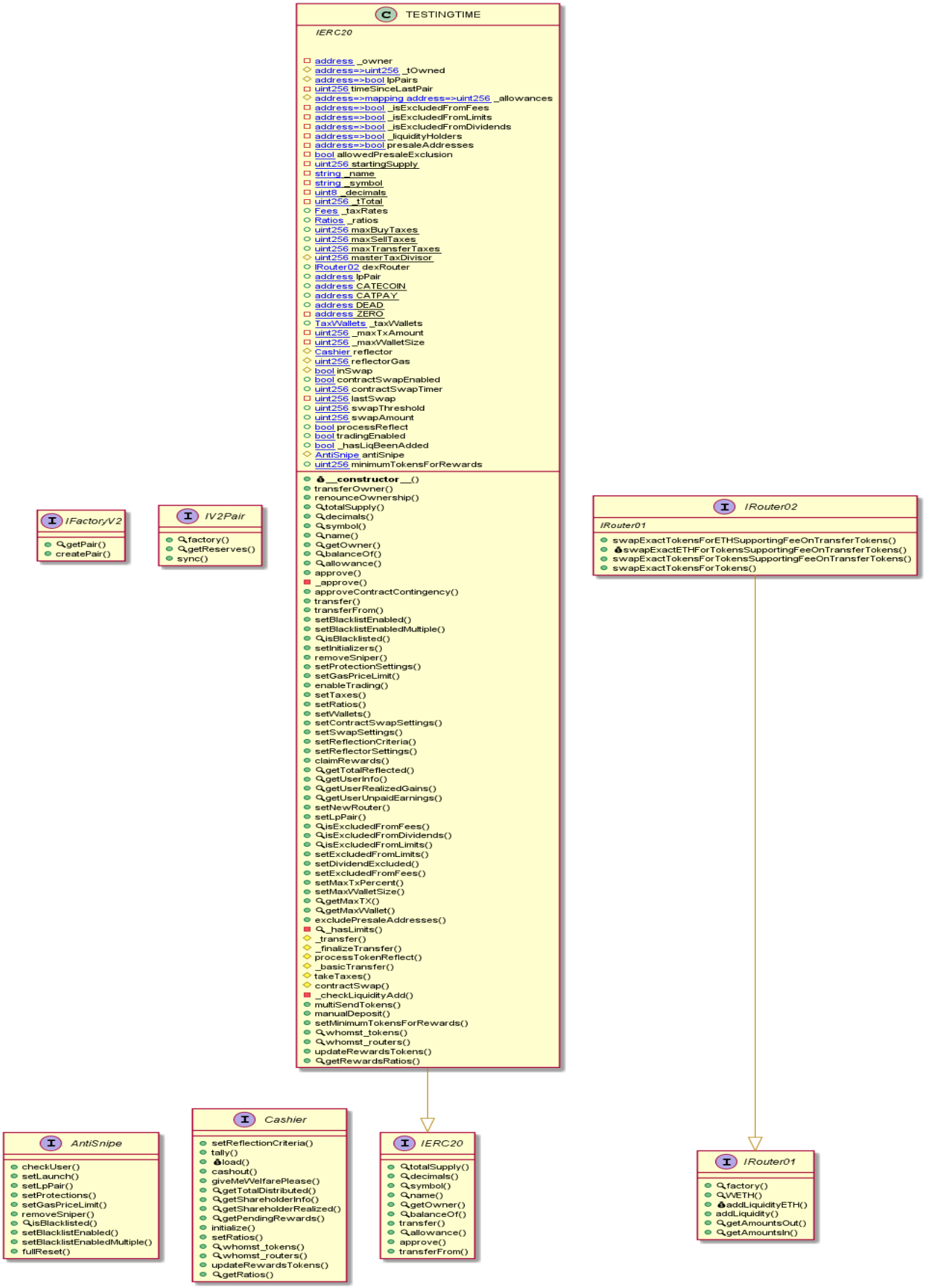
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - CateFarm Token



Slither Results Log

Slither log >> CateFarm.sol

```
INFO:Detectors:
TESTINGTIME.setSwapSettings(uint256,uint256,uint256,uint256,uint256) (TESTINGTIME.sol#424-428) should emit an event for:
        - swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor (TESTINGTIME.sol#425)
        - swapAmount = (_tTotal * amountPercent) / amountDivisor (TESTINGTIME.sol#426)
        - contractSwapTimer = time (TESTINGTIME.sol#427)
TESTINGTIME.setReflectorSettings(uint256) (TESTINGTIME.sol#435-438) should emit an event for:
        - reflectorGas = gas (TESTINGTIME.sol#437)
TESTINGTIME.setMaxTxPercent(uint256,uint256) (TESTINGTIME.sol#517-520) should emit an event for:
        - _maxTxAmount = (_tTotal * percent) / divisor (TESTINGTIME.sol#519)
TESTINGTIME.setMaxWalletSize(uint256,uint256) (TESTINGTIME.sol#522-525) should emit an event for:
        - _maxWalletSize = (_tTotal * percent) / divisor (TESTINGTIME.sol#524)
TESTINGTIME.setMinimumTokensForRewards(uint256) (TESTINGTIME.sol#792-794) should emit an event for:
        - minimumTokensForRewards = tokens (TESTINGTIME.sol#793)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Variable 'TESTINGTIME._finalizeTransfer(address,address,uint256,bool,bool,bool,bool).check (TESTINGTIME.sol#631)' in TESTING
TIME._finalizeTransfer(address,address,uint256,bool,bool,bool,bool) (TESTINGTIME.sol#621-653) potentially used before declar
ation: checked = check (TESTINGTIME.sol#632)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in TESTINGTIME._finalizeTransfer(address,address,uint256,bool,bool,bool,bool) (TESTINGTIME.sol#621-653):
        External calls:
        - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
        State variables written after the call(s):
        - _tOwned[from] -= amount (TESTINGTIME.sol#642)
        - amountReceived = takeTaxes(from,amount,buy,sell,other) (TESTINGTIME.sol#645)
                - _tOwned[address(this)] += feeAmount (TESTINGTIME.sol#699)
        - _tOwned[to] += amountReceived (TESTINGTIME.sol#647)
Reentrancy in TESTINGTIME._transfer(address,address,uint256) (TESTINGTIME.sol#565-619):
        External calls:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmt,0,path,address(this),block.timestamp)
 (TESTINGTIME.sol#722-728)
                - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block
.timestamp) (TESTINGTIME.sol#734-741)
                - reflector.load{value: rewardsBalance}() (TESTINGTIME.sol#752)
        - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
```

```
                - reflector.tally(from,_tOwned[from]) (TESTINGTIME.sol#661)
                - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
                - reflector.tally(to,0) (TESTINGTIME.sol#666)
                - reflector.tally(to,_tOwned[to]) (TESTINGTIME.sol#668)
                - reflector.cashout(reflectorGas) (TESTINGTIME.sol#672)
        External calls sending eth:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block
.timestamp) (TESTINGTIME.sol#734-741)
                - reflector.load{value: rewardsBalance}() (TESTINGTIME.sol#752)
                - _taxWallets.marketing.transfer(marketingBalance) (TESTINGTIME.sol#756)
                - _taxWallets.team.transfer(teamBalance) (TESTINGTIME.sol#759)
        State variables written after the call(s):
        - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
                - allowedPresaleExclusion = false (TESTINGTIME.sol#775)
Reentrancy in TESTINGTIME.constructor() (TESTINGTIME.sol#234-270):
        External calls:
        - lpPair = IFactoryV2(dexRouter.factory()).createPair(dexRouter.WETH(),address(this)) (TESTINGTIME.sol#256)
        State variables written after the call(s):
        - _approve(_owner,address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#259)
                - _allowances[sender][spender] = amount (TESTINGTIME.sol#324)
        - _approve(address(this),address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#260)
                - _allowances[sender][spender] = amount (TESTINGTIME.sol#324)
        - _isExcludedFromDividends[_owner] = true (TESTINGTIME.sol#265)
        - _isExcludedFromDividends[lpPair] = true (TESTINGTIME.sol#266)
        - _isExcludedFromDividends[address(this)] = true (TESTINGTIME.sol#267)
        - _isExcludedFromDividends[DEAD] = true (TESTINGTIME.sol#268)
        - _isExcludedFromDividends[ZERO] = true (TESTINGTIME.sol#269)
        - _isExcludedFromFees[_owner] = true (TESTINGTIME.sol#262)
        - _isExcludedFromFees[address(this)] = true (TESTINGTIME.sol#263)
        - isExcludedFromFees[DEAD] = true (TESTINGTIME.sol#264)
        - lpPairs[lpPair] = true (TESTINGTIME.sol#257)
Reentrancy in TESTINGTIME.enableTrading() (TESTINGTIME.sol#378-391):
        External calls:
        - antiSnipe.setLaunch(lpPair,uint32(block.number),uint64(block.timestamp),_decimals) (TESTINGTIME.sol#384)
        - reflector.initialize() (TESTINGTIME.sol#385)
        State variables written after the call(s):
```

```
        - allowedPresaleExclusion = false (TESTINGTIME.sol#388)
        - processReflect = true (TESTINGTIME.sol#387)
        - swapAmount = (balanceOf(lpPair) * 1) / 1000 (TESTINGTIME.sol#390)
        - swapThreshold = (balanceOf(lpPair) * 5) / 10000 (TESTINGTIME.sol#389)
Reentrancy in TESTINGTIME.excludePresaleAddresses(address,address) (TESTINGTIME.sol#535-552):
        External calls:
        - setDividendExcluded(router,true) (TESTINGTIME.sol#549)
                - reflector.tally(holder,0) (TESTINGTIME.sol#507)
                - reflector.tally(holder,_tOwned[holder]) (TESTINGTIME.sol#509)
        - setDividendExcluded(presale,true) (TESTINGTIME.sol#550)
                - reflector.tally(holder,0) (TESTINGTIME.sol#507)
                - reflector.tally(holder,_tOwned[holder]) (TESTINGTIME.sol#509)
        State variables written after the call(s):
        - setDividendExcluded(presale,true) (TESTINGTIME.sol#550)
                - _isExcludedFromDividends[holder] = enabled (TESTINGTIME.sol#505)
Reentrancy in TESTINGTIME.setBlacklistEnabled(address,bool) (TESTINGTIME.sol#345-348):
        External calls:
        - antiSnipe.setBlacklistEnabled(account,enabled) (TESTINGTIME.sol#346)
        - setDividendExcluded(account,enabled) (TESTINGTIME.sol#347)
                - reflector.tally(holder,0) (TESTINGTIME.sol#507)
                - reflector.tally(holder,_tOwned[holder]) (TESTINGTIME.sol#509)
        State variables written after the call(s):
        - setDividendExcluded(account,enabled) (TESTINGTIME.sol#347)
                - _isExcludedFromDividends[holder] = enabled (TESTINGTIME.sol#505)
Reentrancy in TESTINGTIME.setNewRouter(address) (TESTINGTIME.sol#460-471):
        External calls:
        - lpPair = IFactoryV2(_newRouter.factory()).createPair(address(this),_newRouter.WETH()) (TESTINGTIME.sol#464)
        State variables written after the call(s):
        - _approve(address(this),address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#470)
                - _allowances[sender][spender] = amount (TESTINGTIME.sol#324)
        - dexRouter = _newRouter (TESTINGTIME.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in TESTINGTIME._finalizeTransfer(address,address,uint256,bool,bool,bool,bool) (TESTINGTIME.sol#621-653):
        External calls:
        - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
        Event emitted after the call(s):
        - Transfer(from,address(this),feeAmount) (TESTINGTIME.sol#700)
                - amountReceived = takeTaxes(from,amount,buy,sell,other) (TESTINGTIME.sol#645)
Reentrancy in TESTINGTIME._finalizeTransfer(address,address,uint256,bool,bool,bool,bool) (TESTINGTIME.sol#621-653):
        External calls:
        - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
        - processTokenReflect(from,to) (TESTINGTIME.sol#649)
                - reflector.tally(from,0) (TESTINGTIME.sol#659)
                - reflector.tally(from,_tOwned[from]) (TESTINGTIME.sol#661)
                - reflector.tally(to,0) (TESTINGTIME.sol#666)
                - reflector.tally(to,_tOwned[to]) (TESTINGTIME.sol#668)
                - reflector.cashout(reflectorGas) (TESTINGTIME.sol#672)
        Event emitted after the call(s):
        - Transfer(from,to,amountReceived) (TESTINGTIME.sol#651)
Reentrancy in TESTINGTIME._transfer(address,address,uint256) (TESTINGTIME.sol#565-619):
        External calls:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmt,0,path,address(this),block.timestamp)
 (TESTINGTIME.sol#722-728)
                - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block
.timestamp) (TESTINGTIME.sol#734-741)
                - reflector.load{value: rewardsBalance}() (TESTINGTIME.sol#752)
        - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
                - reflector.tally(from,0) (TESTINGTIME.sol#659)
                - reflector.tally(from,_tOwned[from]) (TESTINGTIME.sol#661)
                - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
                - reflector.tally(to,0) (TESTINGTIME.sol#666)
                - reflector.tally(to,_tOwned[to]) (TESTINGTIME.sol#668)
                - reflector.cashout(reflectorGas) (TESTINGTIME.sol#672)
        External calls sending eth:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block
.timestamp) (TESTINGTIME.sol#734-741)
```

```
                - reflector.load{value: rewardsBalance}() (TESTINGTIME.sol#752)
                - _taxWallets.marketing.transfer(marketingBalance) (TESTINGTIME.sol#756)
                - _taxWallets.team.transfer(teamBalance) (TESTINGTIME.sol#759)
        Event emitted after the call(s):
        - ContractSwapEnabledUpdated(true) (TESTINGTIME.sol#776)
                - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
        - Transfer(from,address(this),feeAmount) (TESTINGTIME.sol#700)
                - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
        - Transfer(from,to,amountReceived) (TESTINGTIME.sol#651)
                - _finalizeTransfer(from,to,amount,takeFee,buy,sell,other) (TESTINGTIME.sol#618)
Reentrancy in TESTINGTIME.constructor() (TESTINGTIME.sol#234-270):
        External calls:
        - lpPair = IFactoryV2(dexRouter.factory()).createPair(dexRouter.WETH(),address(this)) (TESTINGTIME.sol#256)
        Event emitted after the call(s):
        - Approval(sender,spender,amount) (TESTINGTIME.sol#325)
                - _approve(address(this),address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#260)
        - Approval(sender,spender,amount) (TESTINGTIME.sol#325)
                - _approve(_owner,address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#259)
Reentrancy in TESTINGTIME.contractSwap(uint256) (TESTINGTIME.sol#705-761):
        External calls:
        - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmt,0,path,address(this),block.timestamp) (TESTIN
GTIME.sol#722-728)
        - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block.timesta
mp) (TESTINGTIME.sol#734-741)
        External calls sending eth:
        - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block.timesta
mp) (TESTINGTIME.sol#734-741)
        Event emitted after the call(s):
        - AutoLiquify(liquidityBalance,toLiquify) (TESTINGTIME.sol#742)
Reentrancy in TESTINGTIME.setNewRouter(address) (TESTINGTIME.sol#460-471):
        External calls:
        - lpPair = IFactoryV2(_newRouter.factory()).createPair(address(this),_newRouter.WETH()) (TESTINGTIME.sol#464)
        Event emitted after the call(s):
        - Approval(sender,spender,amount) (TESTINGTIME.sol#325)
                - _approve(address(this),address(dexRouter),type()(uint256).max) (TESTINGTIME.sol#470)
Reentrancy in TESTINGTIME.transferOwner(address) (TESTINGTIME.sol#277-292):
        External calls:
```

```
        - _finalizeTransfer(_owner,newOwner,balanceOf(_owner),false,false,false,true) (TESTINGTIME.sol#286)
                - reflector.tally(from,0) (TESTINGTIME.sol#659)
                - reflector.tally(from,_tOwned[from]) (TESTINGTIME.sol#661)
                - antiSnipe.checkUser(from,to,amount) (TESTINGTIME.sol#631-635)
                - reflector.tally(to,0) (TESTINGTIME.sol#666)
                - reflector.tally(to,_tOwned[to]) (TESTINGTIME.sol#668)
                - reflector.cashout(reflectorGas) (TESTINGTIME.sol#672)
        Event emitted after the call(s):
        - OwnershipTransferred(_owner,newOwner) (TESTINGTIME.sol#290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
TESTINGTIME.setLpPair(address,bool) (TESTINGTIME.sol#473-485) uses timestamp for comparisons
        Dangerous comparisons:
        - timeSinceLastPair != 0 (TESTINGTIME.sol#478)
        - require(bool,string)(block.timestamp - timeSinceLastPair > 259200,Cannot set a new pair this week!) (TESTINGTIME.s
ol#479)
TESTINGTIME._transfer(address,address,uint256) (TESTINGTIME.sol#565-619) uses timestamp for comparisons
        Dangerous comparisons:
        - lastSwap + contractSwapTimer < block.timestamp (TESTINGTIME.sol#607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
TESTINGTIME.setLpPair(address,bool) (TESTINGTIME.sol#473-485) compares to a boolean constant:
        -enabled == false (TESTINGTIME.sol#474)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
TESTINGTIME._basicTransfer(address,address,uint256) (TESTINGTIME.sol#676-681) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version>=0.6.0<0.9.0 (TESTINGTIME.sol#6) is too complex
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IRouter01.WETH() (TESTINGTIME.sol#37) is not in mixedCase
Function Cashier.whomst_tokens() (TESTINGTIME.sol#116) is not in mixedCase
Function Cashier.whomst_routers() (TESTINGTIME.sol#117) is not in mixedCase
Parameter TESTINGTIME.setProtectionSettings(bool,bool,bool,bool)._antiSnipe (TESTINGTIME.sol#369) is not in mixedCase
Parameter TESTINGTIME.setProtectionSettings(bool,bool,bool,bool)._antiGas (TESTINGTIME.sol#369) is not in mixedCase
Parameter TESTINGTIME.setProtectionSettings(bool,bool,bool,bool)._antiBlock (TESTINGTIME.sol#369) is not in mixedCase
Parameter TESTINGTIME.setProtectionSettings(bool,bool,bool,bool)._algo (TESTINGTIME.sol#369) is not in mixedCase
Parameter TESTINGTIME.setContractSwapSettings(bool,bool)._enabled (TESTINGTIME.sol#419) is not in mixedCase
Parameter TESTINGTIME.setReflectionCriteria(uint256,uint256,uint256)._minPeriod (TESTINGTIME.sol#430) is not in mixedCase
Parameter TESTINGTIME.setReflectionCriteria(uint256,uint256,uint256)._minReflection (TESTINGTIME.sol#430) is not in mixedCas
e
Function TESTINGTIME.whomst_tokens() (TESTINGTIME.sol#796-798) is not in mixedCase
Function TESTINGTIME.whomst_routers() (TESTINGTIME.sol#800-802) is not in mixedCase
Variable TESTINGTIME._tOwned (TESTINGTIME.sol#126) is not in mixedCase
Variable TESTINGTIME._allowances (TESTINGTIME.sol#129) is not in mixedCase
Constant TESTINGTIME.startingSupply (TESTINGTIME.sol#137) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME._name (TESTINGTIME.sol#139) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME._symbol (TESTINGTIME.sol#140) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME._decimals (TESTINGTIME.sol#141) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME._tTotal (TESTINGTIME.sol#143) is not in UPPER_CASE_WITH_UNDERSCORES
Variable TESTINGTIME._taxRates (TESTINGTIME.sol#159-163) is not in mixedCase
Variable TESTINGTIME._ratios (TESTINGTIME.sol#165-171) is not in mixedCase
Constant TESTINGTIME.maxBuyTaxes (TESTINGTIME.sol#173) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME.maxSellTaxes (TESTINGTIME.sol#174) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME.maxTransferTaxes (TESTINGTIME.sol#175) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TESTINGTIME.masterTaxDivisor (TESTINGTIME.sol#176) is not in UPPER_CASE_WITH_UNDERSCORES
Variable TESTINGTIME._taxWallets (TESTINGTIME.sol#192-196) is not in mixedCase
Variable TESTINGTIME._hasLiqBeenAdded (TESTINGTIME.sol#213) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Reentrancy in TESTINGTIME._transfer(address,address,uint256) (TESTINGTIME.sol#565-619):
        External calls:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - _taxWallets.marketing.transfer(marketingBalance) (TESTINGTIME.sol#756)
                - _taxWallets.team.transfer(teamBalance) (TESTINGTIME.sol#759)
        External calls sending eth:
        - contractSwap(contractTokenBalance) (TESTINGTIME.sol#611)
                - dexRouter.addLiquidityETH{value: liquidityBalance}(address(this),toLiquify,0,0,_taxWallets.liquidity,block
.timestamp) (TESTINGTIME.sol#734-741)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (TESTINGTIME
.sol#49) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDe
sired (TESTINGTIME.sol#50)
Variable TESTINGTIME.setInitializers(address,address).aInitializer (TESTINGTIME.sol#358) is too similar to TESTINGTIME.setIn
itializers(address,address).cInitializer (TESTINGTIME.sol#358)
Variable TESTINGTIME.setRatios(uint16,uint16,uint16,uint16,uint16).rewardsToken1 (TESTINGTIME.sol#402) is too similar to TES
TINGTIME.setRatios(uint16,uint16,uint16,uint16,uint16).rewardsToken2 (TESTINGTIME.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
TESTINGTIME.slitherConstructorConstantVariables() (TESTINGTIME.sol#122-812) uses literals with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (TESTINGTIME.sol#183)
TESTINGTIME.slitherConstructorConstantVariables() (TESTINGTIME.sol#122-812) uses literals with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (TESTINGTIME.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - TESTINGTIME.renounceOwnership() (TESTINGTIME.sol#294-299)
approve(address,uint256) should be declared external:
        - TESTINGTIME.approve(address,uint256) (TESTINGTIME.sol#314-318)
approveContractContingency() should be declared external:
        - TESTINGTIME.approveContractContingency() (TESTINGTIME.sol#328-331)
isBlacklisted(address) should be declared external:
        - TESTINGTIME.isBlacklisted(address) (TESTINGTIME.sol#354-356)
enableTrading() should be declared external:
        - TESTINGTIME.enableTrading() (TESTINGTIME.sol#378-391)
setNewRouter(address) should be declared external:
        - TESTINGTIME.setNewRouter(address) (TESTINGTIME.sol#460-471)
isExcludedFromFees(address) should be declared external:
        - TESTINGTIME.isExcludedFromFees(address) (TESTINGTIME.sol#487-489)
isExcludedFromDividends(address) should be declared external:
        - TESTINGTIME.isExcludedFromDividends(address) (TESTINGTIME.sol#491-493)
isExcludedFromLimits(address) should be declared external:
        - TESTINGTIME.isExcludedFromLimits(address) (TESTINGTIME.sol#495-497)
```

```
isExcludedFromLimits(address) should be declared external:
        - TESTINGTIME.isExcludedFromLimits(address) (TESTINGTIME.sol#495-497)
getMaxTX() should be declared external:
        - TESTINGTIME.getMaxTX() (TESTINGTIME.sol#527-529)
getMaxWallet() should be declared external:
        - TESTINGTIME.getMaxWallet() (TESTINGTIME.sol#531-533)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:TESTINGTIME.sol analyzed (8 contracts with 75 detectors), 80 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

CateFarm.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 557:15:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in TESTINGTIME.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 234:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 479:24:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 482:32:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 740:16:

Gas & Economy

Gas costs:

Gas requirement of function TESTINGTIME._taxRates is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 159:4:

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function TESTINGTIME.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 309:4:

Gas costs:

Gas requirement of function TESTINGTIME.allowance is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 312:4:

Gas costs:

Gas requirement of function TESTINGTIME.setReflectionCriteria is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 430:4:

Gas costs:

Gas requirement of function TESTINGTIME.claimRewards is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 440:4:

Gas costs:

Gas requirement of function TESTINGTIME.getRewardsRatios is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 808:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 782:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 10:2:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 307:4:

Miscellaneous

Constant/View/Pure functions:

Cashier.updateRewardsTokens(address,address,address,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 118:4:

Similar variable names:

TESTINGTIME._approve(address,address,uint256) : Variables have very similar names "sender" and "spender". Note: Modifiers are currently not considered by this static analysis.
Pos: 322:16:

Similar variable names:

TESTINGTIME._approve(address,address,uint256) : Variables have very similar names "sender" and "spender". Note: Modifiers are currently not considered by this static analysis.
Pos: 324:20:

No return:

Cashier.getRatios(): Defines a return type but never explicitly returns a value.
Pos: 119:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 504:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 518:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 783:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 748:30:

Solhint Linter

CateFarm.sol

```
CateFarm.sol:6:1: Error: Compiler version >=0.6.0 <0.9.0 does not
satisfy the r semver requirement
CateFarm.sol:37:5: Error: Function name must be in mixedCase
CateFarm.sol:116:5: Error: Function name must be in mixedCase
CateFarm.sol:117:5: Error: Function name must be in mixedCase
CateFarm.sol:122:1: Error: Contract has 31 states declarations but
allowed no more than 15
CateFarm.sol:126:5: Error: Explicitly mark visibility of state
CateFarm.sol:127:5: Error: Explicitly mark visibility of state
CateFarm.sol:129:5: Error: Explicitly mark visibility of state
CateFarm.sol:137:30: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:139:29: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:140:29: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:141:28: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:143:30: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:173:29: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:174:29: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:175:29: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:176:5: Error: Explicitly mark visibility of state
CateFarm.sol:176:22: Error: Constant name must be in capitalized
SNAKE_CASE
CateFarm.sol:201:5: Error: Explicitly mark visibility of state
CateFarm.sol:202:5: Error: Explicitly mark visibility of state
CateFarm.sol:204:5: Error: Explicitly mark visibility of state
CateFarm.sol:214:5: Error: Explicitly mark visibility of state
CateFarm.sol:234:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
CateFarm.sol:304:32: Error: Code contains empty blocks
CateFarm.sol:384:70: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:384:99: Error: Code contains empty blocks
CateFarm.sol:384:108: Error: Code contains empty blocks
CateFarm.sol:385:36: Error: Code contains empty blocks
CateFarm.sol:385:45: Error: Code contains empty blocks
CateFarm.sol:462:9: Error: Variable name must be in mixedCase
CateFarm.sol:479:25: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:482:33: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:557:16: Error: Avoid to use tx.origin
CateFarm.sol:607:52: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:612:36: Error: Avoid to make time-based decisions in
your business logic
```

```
CateFarm.sol:659:46: Error: Code contains empty blocks
CateFarm.sol:659:55: Error: Code contains empty blocks
CateFarm.sol:661:58: Error: Code contains empty blocks
CateFarm.sol:661:67: Error: Code contains empty blocks
CateFarm.sol:666:44: Error: Code contains empty blocks
CateFarm.sol:666:53: Error: Code contains empty blocks
CateFarm.sol:668:54: Error: Code contains empty blocks
CateFarm.sol:668:63: Error: Code contains empty blocks
CateFarm.sol:672:49: Error: Code contains empty blocks
CateFarm.sol:672:58: Error: Code contains empty blocks
CateFarm.sol:683:75: Error: Variable "other" is unused
CateFarm.sol:727:13: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:740:17: Error: Avoid to make time-based decisions in
your business logic
CateFarm.sol:752:57: Error: Code contains empty blocks
CateFarm.sol:752:66: Error: Code contains empty blocks
CateFarm.sol:789:60: Error: Code contains empty blocks
CateFarm.sol:789:69: Error: Code contains empty blocks
CateFarm.sol:796:5: Error: Function name must be in mixedCase
CateFarm.sol:800:5: Error: Function name must be in mixedCase
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io