Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: FutureCoin Token
Platform: Binance Smart Chain
Website: https://e-futurecoin.com
Language: Solidity
Date: November 10th, 2021

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 FutureCoin team to perform the Security audit of the FutureCoin 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 November 10th, 2021.

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

FutureCoin (FTC) is a BEP20 standard token smart contract with other customization like: swapping, adding liquidity, reflation, etc. This audit only considers FutureCoin token smart contract, and does not cover any other smart contracts in the platform.

Audit scope

Name	**Code Review and Security Analysis Report for FutureCoin Token Smart Contract**
Platform	**BSC / Solidity**
File	FUTURECOIN.sol
File MD5 Hash	11703DE04F296142933B28A03B4DB79F
Online code	0xfbec49521e0b65fdd13d21e6d4dfeb97fdf690b2
Audit Date	November 10th, 2021

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: FUTURECOIN • Symbol: FTC • Decimals: 16	**YES, This is valid.**
• Tax Fee : 0.4% • Liquidity Fee: 0.3% • Charity Fee: 0.3% • Maximum Transaction Amount: 0.5 Million FTC • Number Tokens Sell To Add To Liquidity: 50000 FTC	**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, 2 medium and 4 low and some very low level issues. These issues are not critical ones.

Some of the issues related to Centralization Risk and charity wallet token transfer.

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	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	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	Moderated
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**

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 FutureCoin 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 FutureCoin Token.

The FutureCoin 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 FutureCoin Token smart contracts 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.

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	lockTheSwap	modifier	Passed	No Issue
3	name	read	Passed	No Issue
4	symbol	read	Passed	No Issue
5	decimals	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	isExcludedFromReward	read	Passed	No Issue
15	totalFees	read	Passed	No Issue
16	deliver	write	Function input parameters lack of check	Refer Audit Findings
17	reflectionFromToken	read	Passed	No Issue
18	tokenFromReflection	read	Passed	No Issue
19	excludeFromReward	write	access only Owner	No Issue
20	includeInReward	external	Infinite loops possibility	Refer Audit Findings
21	excludeFromFee	write	access only Owner	No Issue
22	includeInFee	write	access only Owner	No Issue
23	setCharityWallet	write	access only Owner	No Issue
24	setMaxTxPercent	external	Function input parameters lack of check	Refer Audit Findings
25	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
26	receive	external	Passed	No Issue
27	removeAllFee	write	Passed	No Issue
28	restoreAllFee	write	Passed	No Issue
29	isExcludedFromFee	read	Passed	No Issue
30	_approve	internal	Passed	No Issue
31	_transfer	internal	Passed	No Issue
32	swapAndLiquify	write	access by lockTheSwap	No Issue

33	swapTokensForEth	write	Passed	No Issue
34	addLiquidity	write	Centralized risk	Refer Audit Findings
35	_tokenTransfer	write	Passed	No Issue
36	_transferBothExcluded	write	Passed	No Issue
37	_transferStandard	write	Passed	No Issue
38	_transferToExcluded	write	Passed	No Issue
39	_transferFromExcluded	write	Passed	No Issue
40	_reflectFee	write	Passed	No Issue
41	_getTValues	read	Passed	No Issue
42	_getRValues	write	Passed	No Issue
43	_getRate	write	Passed	No Issue
44	_getCurrentSupply	read	Infinite loops possibility	Refer Audit Findings
45	_takeLiquidity	write	Passed	No Issue
46	calculateTaxFee	read	Passed	No Issue
47	calculateLiquidityFee	read	Passed	No Issue
48	setTaxFeePercent	write	Function input parameters lack of check, Centralized risk	Refer Audit Findings
49	setLiquidityFeePercent	write	Function input parameters lack of check, Centralized risk	Refer Audit Findings
50	setCharityFeePercentage	write	Function input parameters lack of check, Centralized risk	Refer Audit Findings
51	_msgSender	internal	Passed	No Issue
52	_msgData	internal	Passed	No Issue
53	owner	read	Passed	No Issue
54	onlyOwner	modifier	Passed	No Issue
55	renounceOwnership	write	Possible to gain ownership	Refer Audit Findings
56	transferOwnership	write	access only Owner	No Issue
57	geUnlockTime	read	Passed	No Issue
58	lock	write	Possible to gain ownership	Refer Audit Findings
59	unlock	write	Possible to gain ownership	Refer Audit Findings

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

(1) Centralization Risk:

- Fees can be set to 100, disallowing users to trade.
- setMaxTxPercent can be set to 0 disallowing users to make transactions
- LP- tokens generated by the liquidity fee are not locked. The owner could use them to remove liquidity.

Resolution: It is advised to use a Multi-Sign wallet and store in a safe place Owner's private key.

(2) Tokens cannot be transferred from charity wallet:

Charity wallet get some percentage of fee on each transaction. But that wallet is not able to transfer his token.
Resolution: We suggest allocating tokens to the charity wallet properly.
Status: Ackownelged by auditee.

Low

(1) Infinite loops possibility:
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.

- includeInReward() - _excluded.length.

- _getCurrentSupply() - _excluded.length.

(2) Function input parameters lack of check:

Some functions require validation before execution.

Functions are:

- deliver
- setMaxTxPercent
- setTaxFeePercent
- setLiquidityFeePercent
- setCharityFeePercentage

Resolution: We suggest using validation like for numerical variables that should be greater than 0 and for address type check variables that are not address(0). For percentage type variables, values should have some range like minimum 0 and maximum 100.

(3) Possible to gain ownership:

Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make a contract without the owner but he can regain ownership by following the steps below:

1. Owner calls the lock function in the contract to set the current owner as _previousOwner.
2. Owner calls unlock to unlock the contract and set _owner = _previousOwner.
3. Owner called renounceOwnership to leave the contract without the owner.
4. Owner calls unlock to regain ownership.

Resolution: We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership before calling the lock function.

(4) Gas Efficiency:

When the contract enters the branch `else if (!_isExcluded[sender] && !_isExcluded[recipient])`, the contract will execute the same piece of code _transferStandard(sender, recipient,amount) We recommend removing the following code:

```
//this method is responsible for taking all fee, if takeFee is true
function _tokenTransfer(address sender, address recipient, uint256 amount,bool takeFee) private {
    if(!takeFee)
        removeAllFee();

    if (_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferFromExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
        _transferToExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferStandard(sender, recipient, amount);
    } else if (_isExcluded[sender] && _isExcluded[recipient]) {
        _transferBothExcluded(sender, recipient, amount);
```

Resolution: We suggest removing this code to reduce some gas.

Very Low / Informational / Best practices:

(1) Make variables constant:

```
string private _name = "FUTURECOIN";
string private _symbol = "FTC";
uint8 private _decimals = 16;
```

These variables will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare those variables as constant. Just put a constant keyword.

(2) Visibility can be external over public:

Any functions which are not called internally, should be declared as external. This saves some gas and is considered a good practice.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

(3) Unused event:

```
event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
event SwapAndLiquifyEnabledUpdated(bool enabled);
```

MinTokensBeforeSwapUpdated event is defined but not used in code.

Resolution: We suggest removing unused event.

Centralization

These smart contracts have 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:

- excludeFromReward: Owner can check if the account is already excluded or not.
- includeInReward: Owner can include in reward.
- excludeFromFee: Owner can exclude from fee.
- includeInFee: Owner can include in fee.
- setCharityWallet: Owner can change charity wallet.
- setMaxTxPercent: Owner can set maximum percentage.
- setSwapAndLiquifyEnabled: Owner can set swap and liquify enabled status.
- setTaxFeePercent: Owner can change tax fee.
- setLiquidityFeePercent: Owner can change the liquidity fee.
- setCharityFeePercentage: Owner can change charity fee.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones. 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"**.

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).

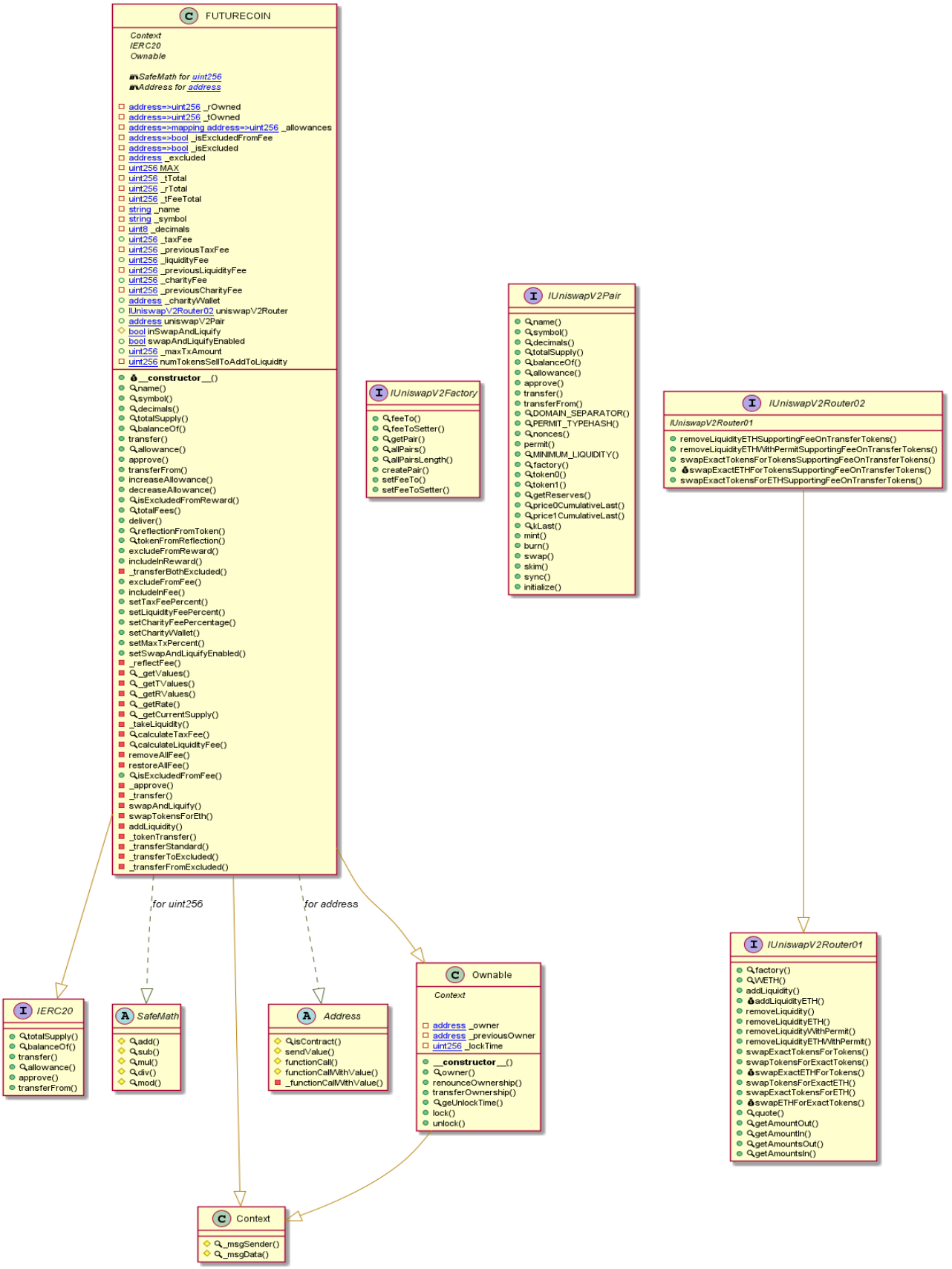
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - FutureCoin Token



Slither Results Log

Slither log >> FUTURECOIN.sol

```
INFO:Detectors:
Reentrancy in FUTURECOIN._transfer(address,address,uint256) (FUTURECOIN.sol#978-1022):
        External calls:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (FUTURECOIN.sol#931)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (FUTURECOIN.sol#1104)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (FUTURECOIN.sol#1115)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (FUTURECOIN.sol#1128)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (FUTURECOIN.sol#831)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (FUTURECOIN.sol#1105)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (FUTURECOIN.sol#1117)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (FUTURECOIN.sol#1129)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (FUTURECOIN.sol#833)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _rTotal = _rTotal.sub(rFee).sub(rCharity) (FUTURECOIN.sol#883)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (FUTURECOIN.sol#933)
                - _tOwned[_charityWallet] = _tOwned[_charityWallet].add(tcharity) (FUTURECOIN.sol#1126)
                - _tOwned[_charityWallet] = _tOwned[_charityWallet].add(tcharity) (FUTURECOIN.sol#829)
                - _tOwned[_charityWallet] = _tOwned[_charityWallet].add(tcharity) (FUTURECOIN.sol#1103)
                - _tOwned[_charityWallet] = _tOwned[_charityWallet].add(tcharity) (FUTURECOIN.sol#1114)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (FUTURECOIN.sol#1127)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (FUTURECOIN.sol#830)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (FUTURECOIN.sol#1116)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (FUTURECOIN.sol#832)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

```
INFO:Detectors:
FUTURECOIN.addLiquidity(uint256,uint256) (FUTURECOIN.sol#1065-1078) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmo
unt}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.sol#1070-1077)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FUTURECOIN.allowance(address,address).owner (FUTURECOIN.sol#746) shadows:
        - Ownable.owner() (FUTURECOIN.sol#388-390) (function)
FUTURECOIN._approve(address,address,uint256).owner (FUTURECOIN.sol#970) shadows:
        - Ownable.owner() (FUTURECOIN.sol#388-390) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.constructor().msgSender (FUTURECOIN.sol#380) lacks a zero-check on :
                - _owner = msgSender (FUTURECOIN.sol#381)
FUTURECOIN.constructor(address).charityWallet (FUTURECOIN.sol#700) lacks a zero-check on :
                - _charityWallet = charityWallet (FUTURECOIN.sol#714)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FUTURECOIN._transfer(address,address,uint256) (FUTURECOIN.sol#978-1022):
        External calls:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _charityFee = _previousCharityFee (FUTURECOIN.sol#963)
                - _charityFee = 0 (FUTURECOIN.sol#957)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _liquidityFee = _previousLiquidityFee (FUTURECOIN.sol#962)
                - _liquidityFee = 0 (FUTURECOIN.sol#956)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _previousCharityFee = _charityFee (FUTURECOIN.sol#953)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
```

```
                - _previousLiquidityFee = _liquidityFee (FUTURECOIN.sol#952)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _previousTaxFee = _taxFee (FUTURECOIN.sol#951)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _tFeeTotal = _tFeeTotal.add(tFee) (FUTURECOIN.sol#884)
        - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
                - _taxFee = _previousTaxFee (FUTURECOIN.sol#961)
                - _taxFee = 0 (FUTURECOIN.sol#955)
Reentrancy in FUTURECOIN.constructor(address) (FUTURECOIN.sol#700-718):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (FUTURECOIN.sol
#705-706)
        State variables written after the call(s):
        - _charityWallet = charityWallet (FUTURECOIN.sol#714)
        - excludeFromReward(charityWallet) (FUTURECOIN.sol#715)
                - _excluded.push(account) (FUTURECOIN.sol#812)
        - excludeFromReward(charityWallet) (FUTURECOIN.sol#715)
                - _isExcluded[account] = true (FUTURECOIN.sol#811)
        - _isExcludedFromFee[owner()] = true (FUTURECOIN.sol#712)
        - _isExcludedFromFee[address(this)] = true (FUTURECOIN.sol#713)
        - excludeFromReward(charityWallet) (FUTURECOIN.sol#715)
                - _tOwned[account] = tokenFromReflection(_rOwned[account]) (FUTURECOIN.sol#809)
        - uniswapV2Router = _uniswapV2Router (FUTURECOIN.sol#709)
Reentrancy in FUTURECOIN.swapAndLiquify(uint256) (FUTURECOIN.sol#1024-1045):
        External calls:
        - swapTokensForEth(half) (FUTURECOIN.sol#1036)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
```

```
        External calls:
        - swapTokensForEth(half) (FUTURECOIN.sol#1036)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - _allowances[owner][spender] = amount (FUTURECOIN.sol#974)
Reentrancy in FUTURECOIN.transferFrom(address,address,uint256) (FUTURECOIN.sol#755-759):
        External calls:
        - _transfer(sender,recipient,amount) (FUTURECOIN.sol#756)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (FUTURECOIN.sol#756)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (FUTURECOI
N.sol#757)
                - _allowances[owner][spender] = amount (FUTURECOIN.sol#974)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in FUTURECOIN._transfer(address,address,uint256) (FUTURECOIN.sol#978-1022):
        External calls:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
```

```
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (FUTURECOIN.sol#1009)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        Event emitted after the call(s):
        - CharityAmount(msg.sender,tcharity) (FUTURECOIN.sol#1109)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - CharityAmount(msg.sender,tcharity) (FUTURECOIN.sol#1121)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - CharityAmount(msg.sender,tcharity) (FUTURECOIN.sol#1133)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - CharityAmount(msg.sender,tcharity) (FUTURECOIN.sol#837)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - Transfer(sender,recipient,tTransferAmount) (FUTURECOIN.sol#1108)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - Transfer(sender,recipient,tTransferAmount) (FUTURECOIN.sol#1132)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - Transfer(sender,recipient,tTransferAmount) (FUTURECOIN.sol#1120)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
        - Transfer(sender,recipient,tTransferAmount) (FUTURECOIN.sol#836)
                - _tokenTransfer(from,to,amount,takeFee) (FUTURECOIN.sol#1021)
Reentrancy in FUTURECOIN.constructor(address) (FUTURECOIN.sol#700-718):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (FUTURECOIN.sol
#705-706)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (FUTURECOIN.sol#717)
Reentrancy in FUTURECOIN.swapAndLiquify(uint256) (FUTURECOIN.sol#1024-1045):
        External calls:
        - swapTokensForEth(half) (FUTURECOIN.sol#1036)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
```

```
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (FUTURECOIN.sol#975)
                - addLiquidity(otherHalf,newBalance) (FUTURECOIN.sol#1042)
        - SwapAndLiquify(half,newBalance,otherHalf) (FUTURECOIN.sol#1044)
Reentrancy in FUTURECOIN.transferFrom(address,address,uint256) (FUTURECOIN.sol#755-759):
        External calls:
        - _transfer(sender,recipient,amount) (FUTURECOIN.sol#756)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (F
UTURECOIN.sol#1056-1062)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (FUTURECOIN.sol#756)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (FUTURECOIN.so
l#1070-1077)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (FUTURECOIN.sol#975)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (F
UTURECOIN.sol#757)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (FUTURECOIN.sol#435-440) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (FUTURECOIN.sol#437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (FUTURECOIN.sol#252-261) uses assembly
        - INLINE ASM (FUTURECOIN.sol#259)
Address._functionCallWithValue(address,bytes,uint256,string) (FUTURECOIN.sol#345-366) uses assembly
        - INLINE ASM (FUTURECOIN.sol#358-361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (FUTURECOIN.sol#345-366) is never used and should be removed
Address.functionCall(address,bytes) (FUTURECOIN.sol#305-307) is never used and should be removed
Address.functionCall(address,bytes,string) (FUTURECOIN.sol#315-317) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (FUTURECOIN.sol#330-332) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (FUTURECOIN.sol#340-343) is never used and should be removed
Address.isContract(address) (FUTURECOIN.sol#252-261) is never used and should be removed
```

```
Address.sendValue(address,uint256) (FUTURECOIN.sol#279-285) is never used and should be removed
Context._msgData() (FUTURECOIN.sol#228-231) is never used and should be removed
SafeMath.mod(uint256,uint256) (FUTURECOIN.sol#201-203) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (FUTURECOIN.sol#217-220) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
FUTURECOIN._rTotal (FUTURECOIN.sol#661) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
FUTURECOIN._previousTaxFee (FUTURECOIN.sol#669) is set pre-construction with a non-constant function or state variable:
        - _taxFee
FUTURECOIN._previousLiquidityFee (FUTURECOIN.sol#672) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
FUTURECOIN._previousCharityFee (FUTURECOIN.sol#675) is set pre-construction with a non-constant function or state variable:
        - _charityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.0 (FUTURECOIN.sol#9) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (FUTURECOIN.sol#279-285):
        - (success) = recipient.call{value: amount}() (FUTURECOIN.sol#283)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (FUTURECOIN.sol#345-366):
        - (success,returndata) = target.call{value: weiValue}(data) (FUTURECOIN.sol#349)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (FUTURECOIN.sol#474) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (FUTURECOIN.sol#475) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (FUTURECOIN.sol#492) is not in mixedCase
Function IUniswapV2Router01.WETH() (FUTURECOIN.sol#512) is not in mixedCase
Parameter FUTURECOIN.setSwapAndLiquifyEnabled(bool)._enabled (FUTURECOIN.sol#874) is not in mixedCase
Parameter FUTURECOIN.calculateTaxFee(uint256)._amount (FUTURECOIN.sol#936) is not in mixedCase
Parameter FUTURECOIN.calculateLiquidityFee(uint256)._amount (FUTURECOIN.sol#942) is not in mixedCase
Variable FUTURECOIN._taxFee (FUTURECOIN.sol#668) is not in mixedCase
Variable FUTURECOIN._liquidityFee (FUTURECOIN.sol#671) is not in mixedCase
Variable FUTURECOIN._charityFee (FUTURECOIN.sol#674) is not in mixedCase
Variable FUTURECOIN._charityWallet (FUTURECOIN.sol#677) is not in mixedCase
Variable FUTURECOIN._maxTxAmount (FUTURECOIN.sol#685) is not in mixedCase
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (FUTURECOIN.sol#229)" inContext (FUTURECOIN.sol#223-232)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (FUTURECOIN.sol#
517) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (F
UTURECOIN.sol#518)
Variable FUTURECOIN._getValues(uint256).rTransferAmount (FUTURECOIN.sol#890) is too similar to FUTURECOIN._transferFromExcluded(address,a
ddress,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._transferStandard(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1102) is too similar to FUTURECOIN._transf
erFromExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._transferToExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1113) is too similar to FUTURECOIN._tran
sferFromExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._transferBothExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#828) is too similar to FUTURECOIN._get
TValues(uint256).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._getValues(uint256).rTransferAmount (FUTURECOIN.sol#890) is too similar to FUTURECOIN._getTValues(uint256).tTransferA
mount (FUTURECOIN.sol#898)
Variable FUTURECOIN.reflectionFromToken(uint256,bool).rTransferAmount (FUTURECOIN.sol#794) is too similar to FUTURECOIN._transferToExclud
ed(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._transferStandard(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1102) is too similar to FUTURECOIN._transf
erToExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._transferStandard(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1102) is too similar to FUTURECOIN._transf
erStandard(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Variable FUTURECOIN._transferToExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1113) is too similar to FUTURECOIN._tran
sferToExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._transferStandard(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1102) is too similar to FUTURECOIN._getTVa
lues(uint256).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._transferToExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1113) is too similar to FUTURECOIN._getT
Values(uint256).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._transferStandard(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1102) is too similar to FUTURECOIN._transf
erBothExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
Variable FUTURECOIN.reflectionFromToken(uint256,bool).rTransferAmount (FUTURECOIN.sol#794) is too similar to FUTURECOIN._transferFromExcl
uded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._transferFromExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1125) is too similar to FUTURECOIN._ge
tTValues(uint256).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._transferBothExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#828) is too similar to FUTURECOIN._tra
nsferBothExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
```

```
Variable FUTURECOIN.reflectionFromToken(uint256,bool).rTransferAmount (FUTURECOIN.sol#794) is too similar to FUTURECOIN._getTValues(uint2
56).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._transferFromExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1125) is too similar to FUTURECOIN._tr
ansferFromExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._getValues(uint256).rTransferAmount (FUTURECOIN.sol#890) is too similar to FUTURECOIN._transferToExcluded(address,add
ress,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._transferBothExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#828) is too similar to FUTURECOIN._tra
nsferToExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._transferBothExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#828) is too similar to FUTURECOIN._tra
nsferFromExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._transferBothExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#828) is too similar to FUTURECOIN._tra
nsferStandard(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Variable FUTURECOIN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (FUTURECOIN.sol#907) is too similar to FUTURECOI
N._transferStandard(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Variable FUTURECOIN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (FUTURECOIN.sol#907) is too similar to FUTURECOI
N._transferToExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN._getValues(uint256).rTransferAmount (FUTURECOIN.sol#890) is too similar to FUTURECOIN._transferBothExcluded(address,a
ddress,uint256).tTransferAmount (FUTURECOIN.sol#828)
Variable FUTURECOIN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (FUTURECOIN.sol#907) is too similar to FUTURECOI
N._transferFromExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1125)
Variable FUTURECOIN._getValues(uint256).rTransferAmount (FUTURECOIN.sol#890) is too similar to FUTURECOIN._transferStandard(address,addre
ss,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Variable FUTURECOIN._transferToExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1113) is too similar to FUTURECOIN._tran
sferBothExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
Variable FUTURECOIN._transferFromExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1125) is too similar to FUTURECOIN._tr
ansferBothExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
Variable FUTURECOIN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (FUTURECOIN.sol#907) is too similar to FUTURECOI
N._transferBothExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
Variable FUTURECOIN._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (FUTURECOIN.sol#907) is too similar to FUTURECOI
N._getTValues(uint256).tTransferAmount (FUTURECOIN.sol#898)
Variable FUTURECOIN._transferFromExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1125) is too similar to FUTURECOIN._tr
ansferStandard(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Variable FUTURECOIN._transferFromExcluded(address,address,uint256).rTransferAmount (FUTURECOIN.sol#1125) is too similar to FUTURECOIN._tr
ansferToExcluded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1113)
Variable FUTURECOIN.reflectionFromToken(uint256,bool).rTransferAmount (FUTURECOIN.sol#794) is too similar to FUTURECOIN._transferBothExcl
uded(address,address,uint256).tTransferAmount (FUTURECOIN.sol#828)
```

```
Variable FUTURECOIN.reflectionFromToken(uint256,bool).rTransferAmount (FUTURECOIN.sol#794) is too similar to FUTURECOIN._transferStandard
(address,address,uint256).tTransferAmount (FUTURECOIN.sol#1102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
FUTURECOIN.slitherConstructorVariables() (FUTURECOIN.sol#646-1137) uses literals with too many digits:
        - _tTotal = 119700000 * 10 ** 16 (FUTURECOIN.sol#660)
FUTURECOIN.slitherConstructorVariables() (FUTURECOIN.sol#646-1137) uses literals with too many digits:
        - _maxTxAmount = 500000 * 10 ** 16 (FUTURECOIN.sol#685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
FUTURECOIN._decimals (FUTURECOIN.sol#666) should be constant
FUTURECOIN._name (FUTURECOIN.sol#664) should be constant
FUTURECOIN._symbol (FUTURECOIN.sol#665) should be constant
FUTURECOIN._tTotal (FUTURECOIN.sol#660) should be constant
FUTURECOIN.numTokensSellToAddToLiquidity (FUTURECOIN.sol#686) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (FUTURECOIN.sol#407-410)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (FUTURECOIN.sol#416-420)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (FUTURECOIN.sol#422-424)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (FUTURECOIN.sol#427-432)
unlock() should be declared external:
        - Ownable.unlock() (FUTURECOIN.sol#435-440)
name() should be declared external:
        - FUTURECOIN.name() (FUTURECOIN.sol#720-722)
symbol() should be declared external:
        - FUTURECOIN.symbol() (FUTURECOIN.sol#724-726)
decimals() should be declared external:
        - FUTURECOIN.decimals() (FUTURECOIN.sol#728-730)
totalSupply() should be declared external:
        - FUTURECOIN.totalSupply() (FUTURECOIN.sol#732-734)
transfer(address,uint256) should be declared external:
        - FUTURECOIN.transfer(address,uint256) (FUTURECOIN.sol#741-744)
allowance(address,address) should be declared external:
```

```
totalSupply() should be declared external:
        - FUTURECOIN.totalSupply() (FUTURECOIN.sol#732-734)
transfer(address,uint256) should be declared external:
        - FUTURECOIN.transfer(address,uint256) (FUTURECOIN.sol#741-744)
allowance(address,address) should be declared external:
        - FUTURECOIN.allowance(address,address) (FUTURECOIN.sol#746-748)
approve(address,uint256) should be declared external:
        - FUTURECOIN.approve(address,uint256) (FUTURECOIN.sol#750-753)
transferFrom(address,address,uint256) should be declared external:
        - FUTURECOIN.transferFrom(address,address,uint256) (FUTURECOIN.sol#755-759)
increaseAllowance(address,uint256) should be declared external:
        - FUTURECOIN.increaseAllowance(address,uint256) (FUTURECOIN.sol#761-764)
decreaseAllowance(address,uint256) should be declared external:
        - FUTURECOIN.decreaseAllowance(address,uint256) (FUTURECOIN.sol#766-769)
isExcludedFromReward(address) should be declared external:
        - FUTURECOIN.isExcludedFromReward(address) (FUTURECOIN.sol#771-773)
totalFees() should be declared external:
        - FUTURECOIN.totalFees() (FUTURECOIN.sol#775-777)
deliver(uint256) should be declared external:
        - FUTURECOIN.deliver(uint256) (FUTURECOIN.sol#779-786)
reflectionFromToken(uint256,bool) should be declared external:
        - FUTURECOIN.reflectionFromToken(uint256,bool) (FUTURECOIN.sol#788-797)
excludeFromFee(address) should be declared external:
        - FUTURECOIN.excludeFromFee(address) (FUTURECOIN.sol#840-842)
includeInFee(address) should be declared external:
        - FUTURECOIN.includeInFee(address) (FUTURECOIN.sol#844-846)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - FUTURECOIN.setSwapAndLiquifyEnabled(bool) (FUTURECOIN.sol#874-877)
isExcludedFromFee(address) should be declared external:
        - FUTURECOIN.isExcludedFromFee(address) (FUTURECOIN.sol#966-968)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:FUTURECOIN.sol analyzed (10 contracts with 75 detectors), 114 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

FUTURECOIN.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 259:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 358:16:

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: 430:20:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 283:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 349:50:

Gas & Economy

Gas costs:

Gas requirement of function FUTURECOIN.lock 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: 427:4:

Email: audit@EtherAuthority.io

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: 919:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 465:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Similar variable names:

FUTURECOIN.(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 701:8:

No return:

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.
Pos: 13:4:

No return:

IERC20.balanceOf(address): Defines a return type but never explicitly returns a value.
Pos: 18:4:

No return:

IERC20.transfer(address,uint256): Defines a return type but never explicitly returns a value.
Pos: 27:4:

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: 93:8:

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: 123:8:

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: 985:8:

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: 987:12:

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: 148:16:

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: 183:20:

Solhint Linter

FUTURECOIN.sol

```
FUTURECOIN.sol:9:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
FUTURECOIN.sol:379:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
FUTURECOIN.sol:430:21: Error: Avoid to make time-based decisions in
your business logic
FUTURECOIN.sol:437:17: Error: Avoid to make time-based decisions in
your business logic
FUTURECOIN.sol:474:5: Error: Function name must be in mixedCase
FUTURECOIN.sol:475:5: Error: Function name must be in mixedCase
FUTURECOIN.sol:492:5: Error: Function name must be in mixedCase
FUTURECOIN.sol:512:5: Error: Function name must be in mixedCase
FUTURECOIN.sol:646:1: Error: Contract has 23 states declarations but
allowed no more than 15
FUTURECOIN.sol:682:5: Error: Explicitly mark visibility of state
FUTURECOIN.sol:700:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
FUTURECOIN.sol:880:32: Error: Code contains empty blocks
FUTURECOIN.sol:1061:13: Error: Avoid to make time-based decisions in
your business logic
FUTURECOIN.sol:1076:13: Error: Avoid to make time-based decisions in
your business logic
```

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