CRYPTO ALERT NFT

BLOCKCHAIN SECURITY ANALYSIS GROUP

FastTruck

Full contract Audit & Code security assessment

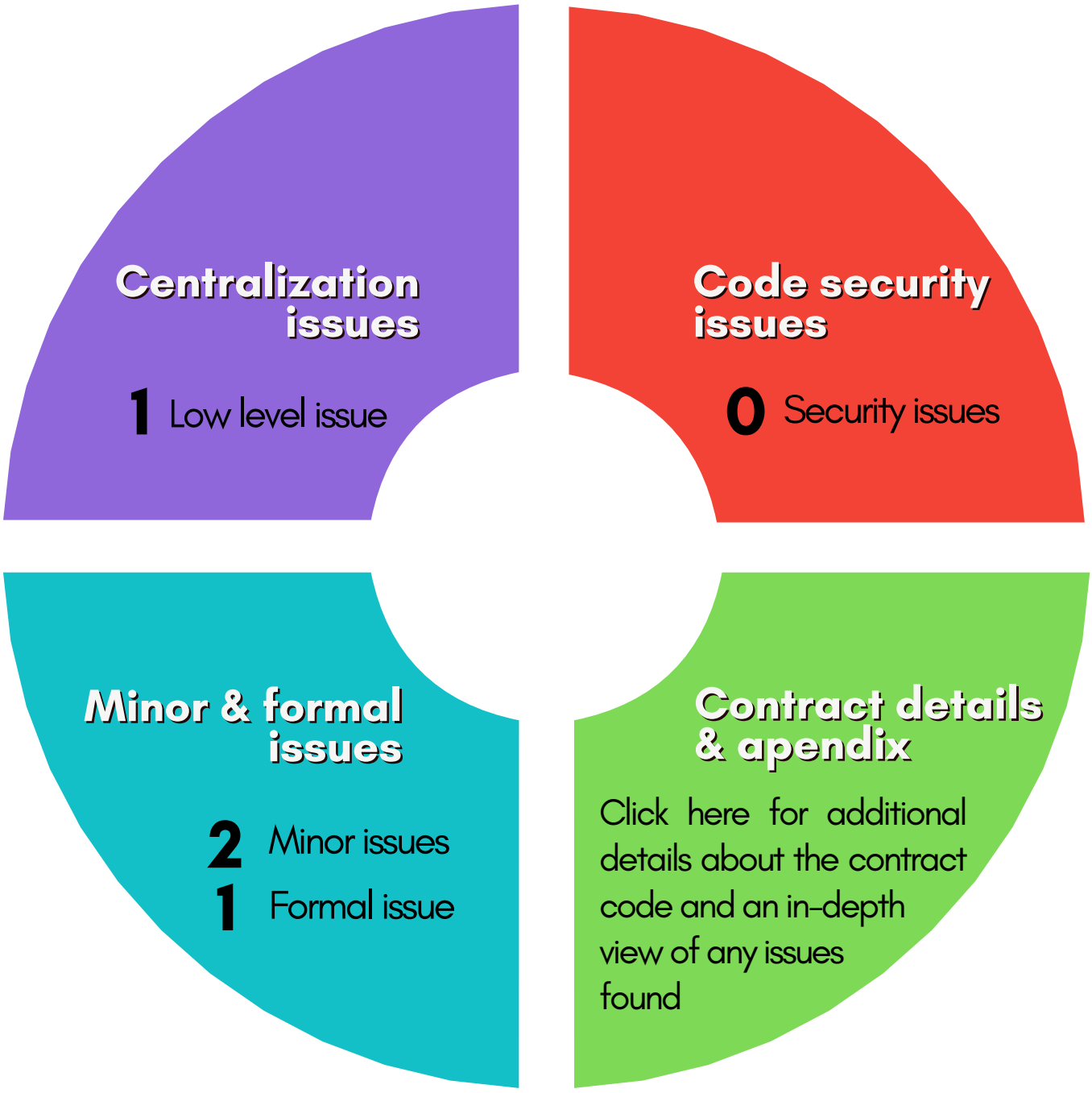
Jun 2023

v0.2

Audit results quick-view and navigation hub

Here you can see an overview of the smart contract code audit

Click on the different colored sections to go to the relevant audit pages for an in-depth report of any issues found.



Centralization issues

The FTCoin.sol contract has a **medium-low** level of centralization.

CE1: Owner can set taxes up to 20%

What does this mean?

Owner can modify buy & sell taxes, while there is a limit to the maximum tax so this function can't be used to make the contract a honeypot, it stills gives some degree of administrative privilege to the owner wallet.

We recommend renouncing ownership after setting up final tax values

Click here or go to page 10 on the technical data appendix to see the relevant contract code and read additional details

Overall centralization status:

Centralized fuction	Status	Comments
Owner **can't** mint?	✅	
Owner **can't** burn?	✅	
Owner **can't** pause trades?	✅	
Owner **can't** set blacklist?	✅	
Owner **can't** change sell tax?	❌	Centralization issue (CE1)
Owner **can't** change buy tax?	❌	Centralization issue (CE1)
Owner **can't** set max wallet size?	✅	
Owner **can't** set max Tx amount?	✅	

Minor and formal issues

There are 2 issues in the formal aspects and coding style in the contract. These don't represent any risk for the security or correct functioning of the contract

MI1 | Function type could be modified to improve gas cost

What does this mean?

Some functions could be declared in a way that can slightly reduce gas usage in some of the contract functions

Click here or go to page 11 on the technical data appendix to see the relevant contract code and read additional details

MI2 | Incorrect event emitted

What does this mean?

When the owner makes a change on the taxes there is a notification message that let users know there have been a change. In this contract one of such messages is mislabeled. This doesn't affect correct contract execution but may lead to confusion when reading the event log

Click here or go to page 12 on the technical data appendix to see the relevant contract code and read additional details

FO1 | Floating pragma

What does this mean?

The contract code does not specify a specific version of the compiler but rather a range of compiler versions that can be used for compilation. Developers should make sure that their contract is created using a particular version of Solidity

Click here or go to page 13 on the technical data appendix to see the relevant contract code and read additional details

Code security issues

After exhaustive analysis by automated tools, AI tools and manual line-by-line code review by our security experts, we didn't find any known security vulnerabilities in the FTCoin.sol Smart Contract

0 Issues found

Technical details and apendix

In the following sections you will find additional details about the contract code as well as an in-depth view of any issues found, including the relevant code sections

Overview

Project details	
Name	Fast Truck
Web	https://www.swishfish.io/
Blockchain	Binance Smart Chain
Environment	EVM
Language	Solidity
Compiler version	0.8.6
Contract's Source	Contract deployed at bsc address: 0x688832a6574eC519D07ae02f62Cc0c0B208fD05a
Contracts Audited	FTCoin.sol
Direct imports (contracts and libraries)	Context.sol, ERC20.sol, IERC20.sol, IERC20Metadata.sol, IUniswapV2Factory.sol, IUniswapV2Router01.sol, IUniswapV2Router02.sol, Ownable.sol, SafeMath.sol
Methodology used	Automated Static analysis Computer assisted code review Manual line-by-line code review Test deployment and stress testing

Variable List

Variable	Scope and additional details
IUniswapV2Router02	Private
uniswapV2Pair	Public
DEAD_WALLET	Public, constant
buyTax	Public
sellTax	Public
projectBuyTax	Public
projectSellTax	Public
takeTax	Public
_isExcludedFromFees	Public, mapping(address => bool)
automatedMarketMakerPairs	Public, mapping(address => bool)

Event List

Events	Emitted?
UpdateUniswapV2Router	✅
ExcludeFromFees	✅
ExcludeMultipleAccountsFromFees	✅
SetAutomatedMarketMakerPair	✅
SetBuyTax	✅
SetSellTax	✅
SetProjectBuyTax	✅
SetProjectSellTax	Minor Issue (MI2)
SetProjectWallet	✅
SetTakeTax	✅

Function list

Type	Name	Results
Constructor	constructor	No issues found
External onlyOwner	updateUniswapV2Router	No issues found
Public onlyOwner	excludeFromFees	Minor Issue (MI1)
Public onlyOwner	excludeMultipleAccountsFromFees	Minor Issue (MI1)
External onlyOwner	setBuyTax	Centralization issue (CE1)
External onlyOwner	setSellTax	Centralization issue (CE1)
External onlyOwner	setProjectBuyTax	Centralization issue (CE1)
External onlyOwner	setProjectSellTax	Centralization issue (CE1)
External onlyOwner	setProjectWallet	No issues found
External onlyOwner	setTakeTax	No issues found
Public onlyOwner	setAutomatedMarketMakerPair	Minor Issue (MI1)
Private	_setAutomatedMarketMakerPair	No issues found
Public view	isExcludedFromFees	No issues found
Internal	_transfer	No issues found

CE1 | Owner can change taxes up to 20%

Functions setBuyTax, setSellTax, setProjectBuyTax and setProjectBuyTax give any wallet with the owner role the privilege to change buy ans sell taxes up to 20% for selling and 20% for buying

```
108     function setProjectBuyTax(uint16 _tax) external onlyOwner {
109         require(_tax <= 100);
110         projectBuyTax = _tax;
111 
112         emit SetProjectBuyTax(_tax);
113     }
114 
115     function setProjectSellTax(uint16 _tax) external onlyOwner {
116         require(_tax <= 100);
117         projectSellTax = _tax;
118 
119         emit SetProjectBuyTax(_tax);
120     }
121 
```

While the limits impose to the function mitigate the risk of abuse, the function still give some amount of administrative privilege to the owner wallet

We recommend setting the tax to an appropriate level and then renouncing the ownership of the contract to avoid any potential abuse of the administrative privileges

Status: Acknowledged:

Since this issue doesn't represent a significant threat to the security of the project, contract or users, the team has decided to keep current implementation

MI1 | Function type could be modified to improve gas cost

Since functions *excludeFromFees*, *setAutomatedMarketMakerPair*, and *isExcludedFromFees*, have the *onlyOwner* require they should have been set to external instead of public. Public functions require slightly less gas and therefore can improve overall contract's gas usage

```
77 ▸    function excludeFromFees(address account, bool excluded) public onlyOwner {…}
82
83      function excludeMultipleAccountsFromFees(
84          address[] calldata accounts,
85          bool excluded
86 ▸    ) public onlyOwner {…}
93
94 ▸    function setBuyTax(uint16 _tax) external onlyOwner {…}
100
101 ▸   function setSellTax(uint16 _tax) external onlyOwner {…}
107
108 ▸   function setProjectBuyTax(uint16 _tax) external onlyOwner {…}
114
115 ▸   function setProjectSellTax(uint16 _tax) external onlyOwner {…}
121
122 ▸   function setProjectWallet(address _wallet) external onlyOwner {…}
127
128 ▸   function setTakeTax(bool _value) external onlyOwner {…}
133
134     function setAutomatedMarketMakerPair(address pair, bool value)
135         public
136         onlyOwner
137 ▸   {…}
145
146 ▸   function _setAutomatedMarketMakerPair(address pair, bool value) private {…}
155
156 ▸   function isExcludedFromFees(address account) public view returns (bool) {…}
```

Status: Acknowledged:

Since these functions are rarely needed and the gas consumption difference is not significant at current BNB chain gas prices, the team has decided to keep current implementation

MI2 | Incorrect event emitted

The function *setProjectSellTax* incorrectly emits the event *SetProjectBuyTax* on line 119.
Instead it should have been set to emit the corresponding event *SetProjectSellTax*

```
108     function setProjectBuyTax(uint16 _tax) external onlyOwner {
109         require(_tax <= 100);
110         projectBuyTax = _tax;
111 
112         emit SetProjectBuyTax(_tax);
113     }
114 
115     function setProjectSellTax(uint16 _tax) external onlyOwner {
116         require(_tax <= 100);
117         projectSellTax = _tax;
118 
119         emit SetProjectBuyTax(_tax);
120     }
121 
```

This doesn't affect the correct execution of the contract or its functions, but leaves users without an event notification when sell taxes are changed.
Users should be appropriately informed of any changes to the sell tax via the project media channels

Status: Acknowledged:

Since this issue doesn't represent a significant threat to the security of the project, contract or users, the team has decided to keep current implementation

FO1 | Floating pragma

The contract code does not specify an strict version of the compiler but rather a range of compiler versions that can be used for compilation.

File 1 of 10: FTCoin.sol

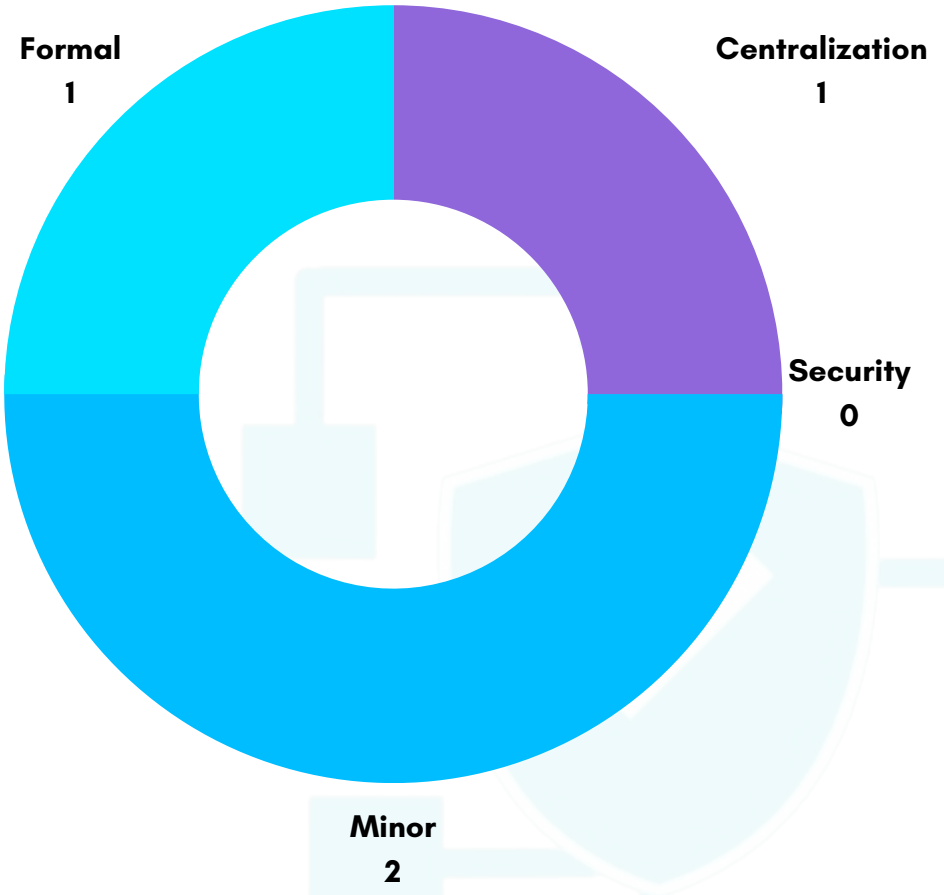
```
1   // SPDX-License-Identifier: MIT
2
3   import "./Ownable.sol";
4   import "./ERC20.sol";
5   import "./IUniswapV2Router02.sol";
6   import "./IUniswapV2Factory.sol";
7
8   pragma solidity ^0.8.6;
9
10  contract FTCoin is ERC20, Ownable {...}
229
```

Developers should make sure that their contract is created using a particular version of Solidity to avoid unexpected bugs that might potentially appear in case different compilers were used in development and in deployment

Status: Acknowledged:

Since this issue doesn't represent a significant threat to the security of the project, contract or users, the team has decided to keep current implementation

Findings summary



Formal: Comments on coding style and best practices issues. Mostly subjective.

Minor: Low-risk issues that cannot harm the contract's execution or expected behaviour.

Code Security High-risk issues that can seriously harm the contract or compromise the ecosystem's security.

Centralization: Excess of centralized privilege can potentially represent an unfair playground for holders and investors.

FTCoin.sol contract findings:

Severity Level	Found	Objected	Acknowledged	Mitigated	Resolved
Formal	1	0	1	0	0
Minor	2	0	2	0	0
Centralization	1	0	1	0	0
Critical	0	0	0	0	0

Synopsis

On May 23th 2023 the Fast Truck development team formally requested our services to evaluate its smart contract for vulnerabilities or issues, as well as the soundness of its code architecture.

Our analysis unit conducted both automated and manual analysis, including line-by-line examination by our team of coding experts, checking the whole code for potential vulnerabilities, centralization issues, unused/redundant code, optimization opportunities, compiling errors, timestamp/order dependencies, reentrancy errors, known attack vectors and coding best practices.

No critical issues or vulnerabilities were found in the code of the smart contracts analized. Some minor issues and certain degree of centralized privileges were detected.

This document describes in detail any findings as well as the mitigating measures taken by the team or those already present in the code (when applicable).

.

Final considerations

In our analysis of the FTCoin.sol token contract we found no security issues.
While the code doesn't seems to have any vulnerabilities that could be exploited by 3rd parties, there are some minor centralization issues.

However, all centralized functions with *onlyOwner* require have limited scope of action, as the owner can't mint additional tokens or pause trading. This significantly reduces the risk of abuse by 3rd parties or external malicious actors. It is still necessary to actively guard the backend from hacking vectors to prevent any potential exploits.

Version history

Version	Changelog
v0.1 (internal)	• Initial review of contract FTCoin.sol
v0.2 First Public Release	• Discussion of results with development team • Amendments to centralization issue CE1 text • Emission of Audit certificate

Disclaimer

The information here provided is not, and must not be considered, endorsement, approval or disapproval of the audited project.
This audit report DOES NOT CONSTITUTE INVESTMENT ADVICE. Its scope is limited to the technical and centralization aspects of the submitted Smart Contracts.
Our team HAS NOT MADE ANY EVALUATION of the project's viability or economic design. Neither do we make any claims about the development team's ability, proficiency or well meaning.
The scope of this review DOES NOT INCLUDE neither the dapp nor the backend of any web3 included in the project design that interacts with the analyzed smart contract, and in consequence we cannot assure their security.
This audit does not constitute any warranty of the absolute bug-free nature of the code or associated technologies used. Neither can it foresee any unwanted results of its interaction with 3rd party solutions like exchanges, in-game databases, or others.
Crypto Alert NFT is in NO WAY RESPONSIBLE for any harm, malfunction or asset loss you might incur while interacting with the smart contracts here evaluated, nor is in any way liable for any detrimental results from your interaction with any aspects of the evaluated project.

Jun 2023

CRYPTO ALERT NFT

BLOCKCHAIN SECURITY ANALYSIS GROUP

https://t.me/cryptoalertnft (community)

https://www.cryptoalertnft.com/ (web)