Guard

Smart contracts security assessment

Final report

Tariff: Standard

FastTruckCoin

February 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

Standard ERC20 token contract. The code is available in the Github repository. The code was checked in the 9c6722f commit. The files Context.sol, ERC20.sol, IERC20.sol, IERC20Metadata.sol, Ownable.sol, Safemath.sol are written according to the OpenZeppellin library standard, which is considered the best practice.

The md5 sum of the file FTCoin.sol - f10b01bc400ddbc2d9ff3076632fff2b.

Name	FastTruckCoin
Audit date	2022-02-22 - 2022-02-24
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
FTCoin	https://github.com/FastTruckUs/FastTruckCoin/blob/9c6722f8d4f4b06147e655af7166c608f7a73546/FTCoin.sol

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed

Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Fee changes are not regulated (FTCoin)

The setBuyFee() and setSellFee() functions do not regulate fee changes, which can lead to the fact that if the owner is compromised, this can lead to a contract failure. For example, if you set buyBurnTax to 110, then on 142L there will be a permanent failure of the contract.

Recommendation: Restrict the inputs in the setBuyFee() and setSellFee() functions via require(). Check that _fee falls within a particular range of values (for example, from 5 to 20).

Medium severity issues

1. No passing 0 values to transfer() (FTCoin)

According to the ERC20 standard(link), tokens must be able to pass a 0 value in the transfer() and transferFrom() functions.

Recommendation: It is recommended to add handling of the 0 value to the transfer() function.

Low severity issues

1. A public modifier must be set for the variables (FTCoin)

Variables 20-22L should have a public modifier.

Recommendation: For transparency of users' work with the platform, it is recommended to make these variables with a public modifier.

2. Not enough events (FTCoin)

Many functions from contract lack events:

1) setBuyFee()

2) setSellFee()

3) transfer()

Recommendation: It is recommended to create events for these functions to ensure transparency of operations.

3. Floating Pragma (FTCoin)

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation: Recommendation: Lock the pragma version and also consider known bugs (link) for the compiler version that is chosen.

4. Naming convention (FTCoin)

Constants should be named with all capital letters with underscores separating words(link). The deadWallet constant is not named by convention.

Recommendation: It is recommended to change the name of the deadWallet constant. This improves the readability of the code.

5. Unnecessary Safemath library (FTCoin)

Since Solidity 0.8, the overflow/underflow check is implemented on the language level - it adds the validation to the bytecode during compilation. You don't need the SafeMath library for Solidity 0.8+(link). Because of this, excess gas will be consumed during deployment and some function calls.

Recommendation: It is recommended to remove the interaction with the SafeMath library.

Conclusion

FastTruckCoin FTCoin contract was audited. 1 high, 1 medium, 5 low severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
FTCoin.setBuyFee(uint16) (contracts/FTCoin.sol#88-90) should emit an event for:
        - buyBurnTax = _fee (contracts/FTCoin.sol#89)
FTCoin.setSellFee(uint16) (contracts/FTCoin.sol#92-94) should emit an event for:
        - sellBurnTax = _fee (contracts/FTCoin.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

FTCoin.updateUniswapV2Router(address)._uniswapV2Pair (contracts/FTCoin.sol#66-67) lacks
a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (contracts/FTCoin.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

Reentrancy in FTCoin.constructor() (contracts/FTCoin.sol#37-57):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(addr
ess(this),_uniswapV2Router.WETH()) (contracts/FTCoin.sol#42-43)
        State variables written after the call(s):
        - _mint(owner(),3500000 * 10 ** uint256(decimals())) (contracts/FTCoin.sol#56)
                - _balances[account] += amount (contracts/ERC20.sol#299)
        - excludeFromFees(owner(),true) (contracts/FTCoin.sol#53)
                - _isExcludedFromFees[account] = excluded (contracts/FTCoin.sol#72)
        - excludeFromFees(address(this),true) (contracts/FTCoin.sol#54)
                - _isExcludedFromFees[account] = excluded (contracts/FTCoin.sol#72)
        - _mint(owner(),3500000 * 10 ** uint256(decimals())) (contracts/FTCoin.sol#56)
                - _totalSupply += amount (contracts/ERC20.sol#298)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (contracts/FTCoin.sol#51)
                - automatedMarketMakerPairs[pair] = value (contracts/FTCoin.sol#113)
        - buyBurnTax = 5 (contracts/FTCoin.sol#48)
        - sellBurnTax = 10 (contracts/FTCoin.sol#49)
        - uniswapV2Pair = _uniswapV2Pair (contracts/FTCoin.sol#46)
        - uniswapV2Router = _uniswapV2Router (contracts/FTCoin.sol#45)
Reentrancy in FTCoin.updateUniswapV2Router(address) (contracts/FTCoin.sol#59-69):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(addre
ss(this),uniswapV2Router.WETH()) (contracts/FTCoin.sol#66-67)
        State variables written after the call(s):
```

```
        - uniswapV2Pair = _uniswapV2Pair (contracts/FTCoin.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in FTCoin.constructor() (contracts/FTCoin.sol#37-57):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(addr
ess(this),_uniswapV2Router.WETH()) (contracts/FTCoin.sol#42-43)
        Event emitted after the call(s):
        - ExcludeFromFees(account,excluded) (contracts/FTCoin.sol#74)
                - excludeFromFees(address(this),true) (contracts/FTCoin.sol#54)
        - ExcludeFromFees(account,excluded) (contracts/FTCoin.sol#74)
                - excludeFromFees(owner(),true) (contracts/FTCoin.sol#53)
        - SetAutomatedMarketMakerPair(pair,value) (contracts/FTCoin.sol#115)
                - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (contracts/
FTCoin.sol#51)
        - Transfer(address(0),account,amount) (contracts/ERC20.sol#300)
                - _mint(owner(),3500000 * 10 ** uint256(decimals())) (contracts/
FTCoin.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.2', '^0.8.0', '^0.8.6']
        - ^0.8.0 (contracts/Context.sol#3)
        - ^0.8.0 (contracts/ERC20.sol#3)
        - ^0.8.6 (contracts/FTCoin.sol#9)
        - ^0.8.0 (contracts/IERC20.sol#3)
        - ^0.8.0 (contracts/IERC20Metadata.sol#3)
        - >=0.5.0 (contracts/IUniswapV2Factory.sol#5)
        - >=0.6.2 (contracts/IUniswapV2Router01.sol#5)
        - >=0.6.2 (contracts/IUniswapV2Router02.sol#5)
        - ^0.8.0 (contracts/Ownable.sol#3)
        - ^0.8.0 (contracts/SafeMath.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Context._msgData() (contracts/Context.sol#20-22) is never used and should be removed
ERC20._burn(address,uint256) (contracts/ERC20.sol#316-331) is never used and should be
removed
SafeMath.add(uint256,uint256) (contracts/SafeMath.sol#111-113) is never used and should
```

```
be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#209-218) is never used and
should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#169-171) is never used and should
be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#235-244) is never used and
should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#186-195) is never used and
should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/SafeMath.sol#20-30) is never used and
should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/SafeMath.sol#74-83) is never used and
should be removed
SafeMath.tryMod(uint256,uint256) (contracts/SafeMath.sol#90-99) is never used and
should be removed
SafeMath.tryMul(uint256,uint256) (contracts/SafeMath.sol#53-67) is never used and
should be removed
SafeMath.trySub(uint256,uint256) (contracts/SafeMath.sol#37-46) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/Context.sol#3) allows old versions
Pragma version^0.8.0 (contracts/ERC20.sol#3) allows old versions
Pragma version^0.8.0 (contracts/IERC20.sol#3) allows old versions
Pragma version^0.8.0 (contracts/IERC20Metadata.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/IUniswapV2Factory.sol#5) allows old versions
Pragma version>=0.6.2 (contracts/IUniswapV2Router01.sol#5) allows old versions
Pragma version>=0.6.2 (contracts/IUniswapV2Router02.sol#5) allows old versions
Pragma version^0.8.0 (contracts/Ownable.sol#3) allows old versions
Pragma version^0.8.0 (contracts/SafeMath.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Parameter FTCoin.setBuyFee(uint16)._fee (contracts/FTCoin.sol#88) is not in mixedCase
Parameter FTCoin.setSellFee(uint16)._fee (contracts/FTCoin.sol#92) is not in mixedCase
Constant FTCoin.deadWallet (contracts/FTCoin.sol#17-18) is not in
UPPER_CASE_WITH_UNDERSCORES
Function IUniswapV2Router01.WETH() (contracts/IUniswapV2Router01.sol#10) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
```

```
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountADesired (contracts/IUniswapV2Router01.sol#15) is too similar
to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,addre
ss,uint256).amountBDesired (contracts/IUniswapV2Router01.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

FTCoin.constructor() (contracts/FTCoin.sol#37-57) uses literals with too many digits:
        - _mint(owner(),3500000 * 10 ** uint256(decimals())) (contracts/FTCoin.sol#56)
FTCoin.slitherConstructorConstantVariables() (contracts/FTCoin.sol#11-159) uses
literals with too many digits:
        - deadWallet = address(0x000000000000000000000000000000000000dEaD) (contracts/
FTCoin.sol#17-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

name() should be declared external:
        - ERC20.name() (contracts/ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (contracts/ERC20.sol#68-70)
totalSupply() should be declared external:
        - ERC20.totalSupply() (contracts/ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (contracts/ERC20.sol#99-107)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (contracts/ERC20.sol#117-125)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (contracts/ERC20.sol#130-138)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (contracts/ERC20.sol#147-155)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (contracts/ERC20.sol#170-187)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (contracts/ERC20.sol#201-212)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (contracts/ERC20.sol#228-243)
excludeMultipleAccountsFromFees(address[],bool) should be declared external:
        - FTCoin.excludeMultipleAccountsFromFees(address[],bool) (contracts/
FTCoin.sol#77-86)
setAutomatedMarketMakerPair(address,bool) should be declared external:
```

```
        - FTCoin.setAutomatedMarketMakerPair(address,bool) (contracts/
FTCoin.sol#96-106)
isExcludedFromFees(address) should be declared external:
        - FTCoin.isExcludedFromFees(address) (contracts/FTCoin.sol#118-120)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/Ownable.sol#56-58)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/Ownable.sol#64-70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (10 contracts with 77 detectors), 50 result(s) found
```

0x Guard