auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

RedFroggy

DATED : 29 September 23'

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

AUDIT SUMMARY

Project name – RedFroggy

Date: 29 September 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	1	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xBbe738927593Ad0c6931C49bf53161ee65093A3d#code

Token Information

Token Address :
0x72D1f1E58a548FF8982a8A64829B40E2e3AF8eb0

Name: RedFroggy

Symbol: Froggy

Decimals: 18

Network: Binance smart chain

Token Type: BEP20

Owner: 0x1A714698dbAF78ED9D4ebE5dBC49851C6dd58454

Deployer:
0x1A714698dbAF78ED9D4ebE5dBC49851C6dd58454

Token Supply: 1,000,000

Checksum:
af747e29e250fa2181f56bced993ee804a62665c

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xBbe738927593Ad0c6931C49bf53161ee65093A3d#code

TOKEN OVERVIEW

buy fee: 0-5%

Sell fee: 0-24%

transfer fee: 0-5%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modifying fees
- Enabling trades

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

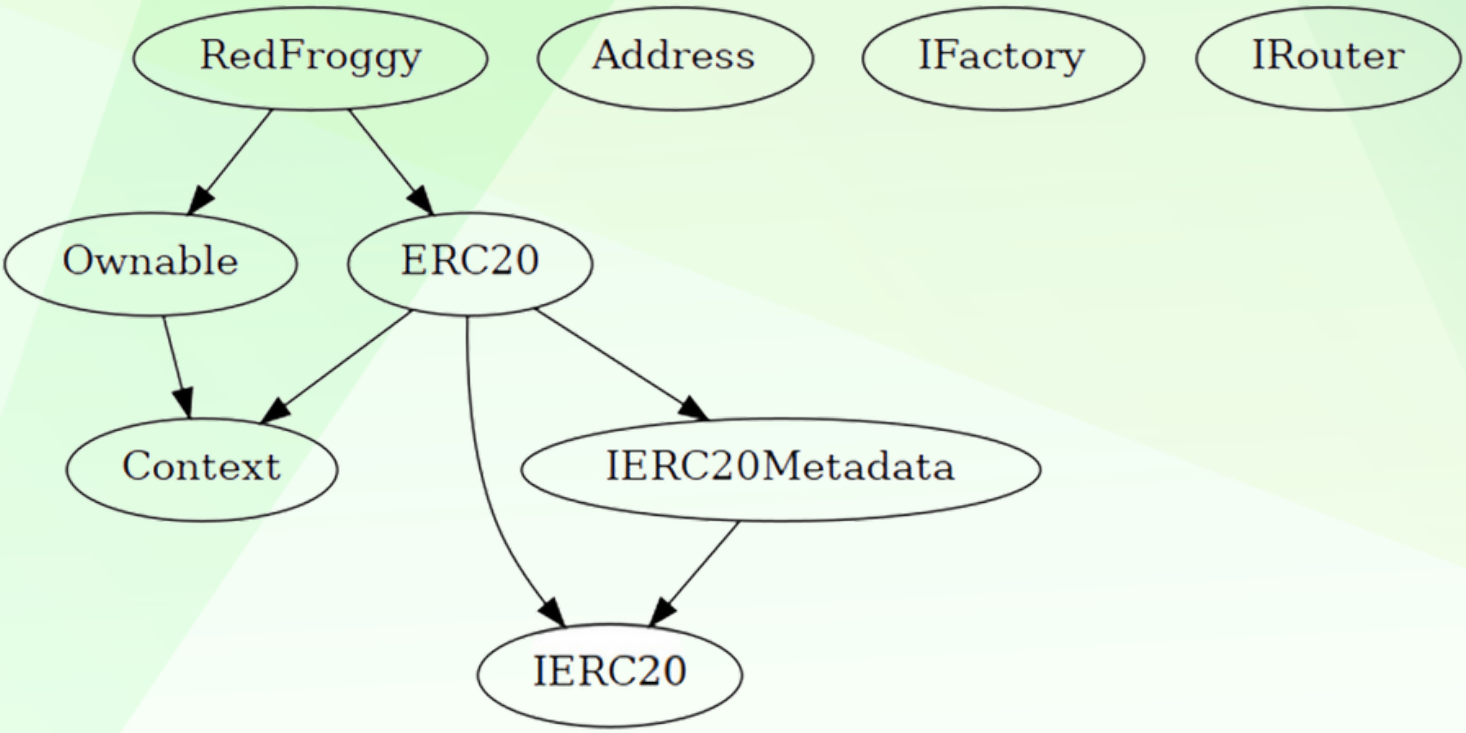
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	1
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is able to adjust buy/transfer fees within 0-5%**
- **Owner is able to adjust sell fees within 0 – 24%**
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens
- Owner is not able to set maximum wallet and maximum buy/sell/transfer limits
- **Owner must enable trades manually**

STATIC ANALYSIS

```
Reentrancy in RedFroggy.transferFrom(address,address,uint256) (contracts/Token.sol#489-504):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#494)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#660-667)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#343)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#646-652)
                - address(devWallet).sendValue(devAmt) (contracts/Token.sol#632)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#494)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#660-667)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#343)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#332)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/Token.sol#501)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (contracts/Token.sol#13-16) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.17 (contracts/Token.sol#6) allows old versions
solc-0.8.17 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) (contracts/Token.sol#337-348):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#343)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable ERC20._balances (contracts/Token.sol#69) is not in mixedCase
Variable ERC20._allowances (contracts/Token.sol#71) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#401) is not in mixedCase
Function RedFroggy.Liquify(uint256,RedFroggy.Taxes) (contracts/Token.sol#596-635) is not in mixedCase
Parameter RedFroggy.updateLiquidityTreshhold(uint256).new_amount (contracts/Token.sol#674) is not in mixedCase
Function RedFroggy.EnableTrading() (contracts/Token.sol#682-687) is not in mixedCase
Parameter RedFroggy.updatedeadline(uint256)._deadline (contracts/Token.sol#689) is not in mixedCase
Parameter RedFroggy.updateExemptFee(address,bool)._address (contracts/Token.sol#718) is not in mixedCase
Variable RedFroggy.genesis_block (contracts/Token.sol#436) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (contracts/Token.sol#14)" inContext (contracts/Token.sol#8-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
RedFroggy.launchtax (contracts/Token.sol#438) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
RedFroggy.pair (contracts/Token.sol#428) should be immutable
RedFroggy.router (contracts/Token.sol#427) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:./contracts/Token.sol analyzed (9 contracts with 88 detectors), 34 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	_tokengeneration	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	

CONTRACT ASSESMENT

Address	Library			
└	sendValue	Internal 🔒	🛑	
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
RedFroggy	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	Liquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForETH	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	updateLiquidityProvide	External ❗	🛑	onlyOwner
└	updateLiquidityTreshhold	External ❗	🛑	onlyOwner
└	EnableTrading	External ❗	🛑	onlyOwner
└	updatedeadline	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateTax	External ❗	🛑	onlyOwner
└	updateExemptFee	External ❗	🛑	onlyOwner
└	bulkExemptFee	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x79faf3c4aaf4a1b7a7f6407a6887e9e1cdc254ac35ad1ae48eccd264cbb48543

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xfbd81aa0058905f4cfb2827589b7f2e3423274fdcb539d40f792d0afeb352262

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5b6c774d6c7e6ce2da5fc4897d88e05161690f591ed148053a6813eb7249c542

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x382c5734be511b4d63536f0641f2fce8108c165c91402b093d56dc28f5bce461

5- Buying when not excluded from fees (tax 0-5%) (passed):
https://testnet.bscscan.com/tx/0xd89db9169852bf7b9b6024a89cf3b6ffc2e2c72abc989f5369f87a9c3664989a

6- Selling when not excluded from fees (tax 0-24%) (passed):
https://testnet.bscscan.com/tx/0xe825a39d60f7fb7eca333a26e1f8165c38d5d27e65e16bb96f4d4564e50c1648

7- Transferring when not excluded from fees (0-5% tax) (passed):
https://testnet.bscscan.com/tx/0x0977feae76704bcd715656c402b015029cdc9d8a6271acb44641970d74ab981d

8- Internal swap (BNB set to dev wallet + Auto-liquidity) (passed):
https://testnet.bscscan.com/tx/0xf91ffae00eeeee82be484ee659aac57433a40ed5388612f9c57254a683812e9b

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

MANUAL TESTING

Logical – Updating swap threshold

Severity: Medium

function: updateLiquidityThreshold

Status: Open

Overview:

updateLiquidityThreshold requires new swap threshold to be less than 1e7 which is equal to 10x of total supply while error message indicates that new swap threshold amount must be less than 1% of total supply (1e5)

```
  function updateLiquidityTreshhold(uint256 new_amount) external
onlyOwner {
    require(
      new_amount <= 1e7,
      "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
  }
```

Suggestion

Change condition to be compatible with the error message:

```
  function updateLiquidityTreshhold(uint256 new_amount) external
onlyOwner {
    require(
      new_amount <= 1e5,
      "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
  }
```

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace