expelee

Building the Futuristic Blockchain Ecosystem

Security Audit Report

FOR

FrogInu

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	Passed With Critical Risk
KYC Verification	Not Done
Audit Date	25 Feb 2023

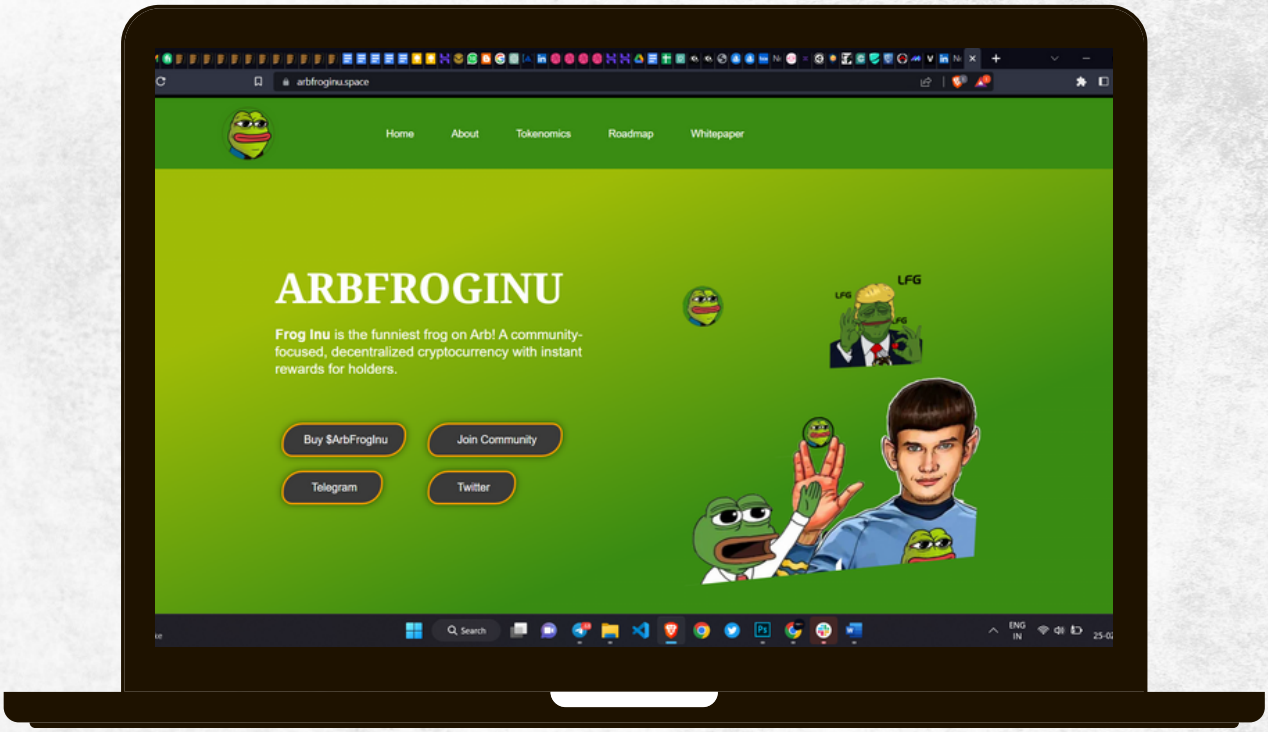
PROJECT DESCRIPTION

FrogInu

ArbFrogInu is a meme token that only rewards investors for holding, the number of rewards received will increase over time if investors hold them longer. We are the safest place for long and short term investors in the meme category. Safemoon x10000 shiba x10000 babydoge x10000 so the next x10000 memecoin will be ArbFrogInu

Social Media Profiles

FrogInu



https://www.arbfroginu.space/

https://t.me/FrogInu_Token

https://twitter.com/Froginutoken

It's always good to check the social profiles of the project, before making your investment.

-Team Expelee

CONTRACT DETAILS

Token Name

FrogInu

Symbol

FrogInu

Network

Arbitrum

Language

Solidity

Contract Address (Verified)

0xDA4d83addc994DD6BC36B47a4DaFDd7E9Adf197d

Token Type

ERC20

Total Supply

1,000,000,000,000,000

Contract SHA-256 Checksum:

202c6a17af70e978f3d061a45ba466950f67e618

Owner Wallet

0x304adBfF27aA8252aAA9B9395B4990F1A2f9f967

Deployer Wallet

0x304adBfF27aA8252aAA9B9395B4990F1A2f9f967

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Complier
- Hardhat

FUNCTION OVERVIEW

Can Take Back Ownership	Not Detected
Owner Change Balance	Not Detected
Blacklist	Detected
Modify Fees	Detected
Proxy	Not Detected
Whitelisted	Not Detected
Anti Whale	Not Detected
Trading Cooldown	Not Detected
Transfer Pausable	Detected
Cannot Sell All	Not Detected
Hidden Owner	Not Detected
Mint	Not Detected

VULNERABILITY CHECKLIST

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

AUDIT SUMMARY

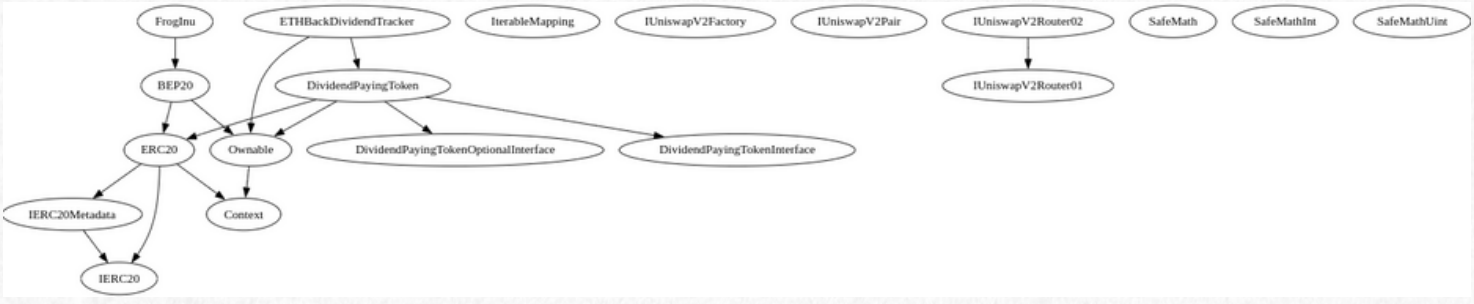
Used Tools:

1.Manual Review: The code has undergone a line-by-line review by the Expelee 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.

Inheritance Trees:



Summary:

- Owner is able to set buy/sell/transfer taxes up to 100%
- Owner is able to disable trades
- Owner is able to blacklist an arbitrary wallet
- Owner is not able to set max tx or max wallet
- Owner is not able to mint new tokens

Functional Tests

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x453ac30cf5bc6a07ff8cca87bc3d6aeea2942fc3c89ed712e646df7612c34694

2- Buying (7% tax – auto burn working) (passed):

https://testnet.bscscan.com/tx/0xbdac3442486a2fa02dac938bcc84af8af812c1cd05b3086c01fb009a5ae2d470

3- Selling (7% tax – auto burn working) (passed):

https://testnet.bscscan.com/tx/0x29eefda68947456e98fd054bfe6a707f9b80ca68bdc43279fb7f637a442036df

4- Transferring (7% tax – auto burn working) (passed):

https://testnet.bscscan.com/tx/0x6a674d07fbe0024368cad4c8fe6d6feab5b2ff30e6276444d00232020101a955

Functional Tests

5- Auto-liquidity (passed):

this is a wallet that received generated LP tokens during a sell
https://testnet.bscscan.com/token/0xbc30dba58911f9e1fa854b5dda913a9745b519fd?a=0x9d54c86d4b34ef6a2dc6352235f4e0f67f501ecf

6- Internal Swap ((passed):

this wallet received WBNB during sell, internal swap can also be checking in above sell transaction
https://testnet.bscscan.com/token/0xed24fc36d5ee211ea25a80239fb8c4cfd80f12ee?a=0x9d54c86d4b34ef6a2dc6352235f4e0f67f501ecf

7- Rewards Auto-Distribution (passed):

this is dividend tracker that sends BUSD rewards to holders after each trade
https://testnet.bscscan.com/token/0xed24fc36d5ee211ea25a80239fb8c4cfd80f12ee?a=0x300416a037c955351d99e5a407e60f648a26f078

MANUAL AUDIT

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into three primary risk categories: high, medium, and low.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

FINDINGS

- Critical Risk Findings:4
- High Risk Findings:0
- Medium Risk Findings:0
- Low Risk Findings:0
- Suggestions & Optimizations 1

Critical Risk Findings

Centralization – Restarting anti-bot at anytime:

The L function allows the owner to disable or enable trades by setting the isL variable to true or false. Additionally, the owner can set the launchB variable to the current block number and specify a value for killNum, which determines the number of blocks after launchB that all buyers will be blacklisted. Blacklisted wallets are then unable to sell or transfer their tokens. This anti-bot mechanism can be restarted by the owner at any time with a new value for killNum and an arbitrary number of dead blocks

```
function L(bool s, uint256 muchB) public onlyOwner {
    isL = s;
    lunachB = block.number;
    killNum = muchB;
}
```

```
if (from == uniswapV2Pair) {
    if (lunachB + killNum > block.number) {
        isbclisted[to] = true;
    }
}
```

Suggestion: To prevent any abuse of the blacklisting feature, it is recommended that the contract implement restrictions to ensure that the launch block number **(launchB)** and number of dead blocks **(killNum)** cannot be modified after the token has been launched. Additionally, the project team should set a reasonable limit on the number of dead blocks, for example, no more than 5 blocks. This will prevent malicious activity and ensure the integrity of the project.

Critical Risk Findings

Centralization – Disabling trades:

Owner is able to disable trades at any given time by setting isL variable to false

```
if (takeFee) {
    require(isL, "ERC20: Transfer not open");
    if (from == uniswapV2Pair) {
        if (lunachB + killNum > block.number) {
            isbclisted[to] = true;
        }
    }
```

```
function L(bool s, uint256 muchB) public onlyOwner {
    isL = s;
    lunachB = block.number;
    killNum = muchB;
}
```

Suggestion: isL must not be changeable after launching.

Centralization – Blacklisting arbitrary wallet(s):
Owner is able to blacklist one or multiple wallets

```
function multi_bclist(
    address[] calldata addresses,
    bool value
) public onlyOwner {
    require(addresses.length < 201);
    for (uint256 i; i < addresses.length; ++i) {
        isbclisted[addresses[i]] = value;
    }
}
```

```
function bclistAddress(address account, bool value) public onlyOwner {
    isbclisted[account] = value;
}
```

Suggestion:
delete this function or declare actions that may lead to getting blacklisted.

Critical Risk Findings

Centralization – Setting fees up to 100%:
Owner is able to set transfer, buy and sell fees up to 100%

Compilation Error - Use of "immutable" keyword in incompatible version:
The "immutable" keyword was introduced in Solidity version 0.8.0 and is not compatible with earlier versions, such as the 0.6.2 compiler used by this contract. This results in a compilation error. We suggest upgrading the Solidity compiler version to 0.8.0 or later to use the "immutable" keyword.

Suggestions & Optimizations:
- use **“indexed”** keyword for event arguments (up to 3)

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 Start-up. Coping up with numerous solutions for blockchain Security and constructing a Web3 Ecosystem from Deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic **Blockchain Ecosystem**

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 Expelee 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 Expelee 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 Expelee team disclaims any liability for the resulting losses.