



Building the Futuristic **Blockchain Ecosystem**

Audit Report FOR





Fight4Hope

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 High Risk
 KYC Verification	Not Done
 Audit Date	19 Sep 2022

Why Passed with High risk?

All the codebase has been reviewed, there was no issues that could disable the system, however there was couple of high risk centralization issues which are discussed in the report

-Team Expelee

PROJECT DESCRIPTION

Fight4Hope

Fight4Hope is a play-to-earn blockchain game based on an online multiplayer battle royale model.

Fight4Hope is built on the Unreal 5 engine with the highest standards in graphics available, never delivered in a crypto play-to-earn concept to date

 **fight4hope.io**

 **fight4hope**

 **fight4hope_**

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

-Team Expelee

CONTRACT DETAILS

Contract Name

Fight4Hope

Token Type

ERC20-ERC 2612

Network

BSC

Language

Solidity

Contract Address (Verified)

0xa5F8128d04692656a60f17B349C2146c48e6863a

Dividend Tracker (Verified)

0xfbe67b7eb6165651beaa82906b0da4cc6fd2a8e1

Compiler

v0.8.11+commit.d7f03943

License

Default license

Contract SHA-256 Checksum:

e48dadecb8af26db506b3219ce4b02da69854478b6337c1be692e4baa13aa4db

Token Name

Fight4Hope

Token Symbol

F4H

Bridge Vault (EOA):

0x1e7E5e693d961E36Ef6DB5C9e4C99C98137F3BB9

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
- Compiler
- Hardhat

FUNCTION OVERVIEW

Can Take Back Ownership	Not Detected
Owner Change Balance	Not Detected
Blacklist	Not Detected
Modify Fees	Not Detected
Proxy	Not Detected
Whitelisted	Not Detected
Anti-Whale	Not Detected
Trading Cooldown	Not Detected
Transfer Pausable	Not 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

Ownership:

current owner of Fight4Hope is:

0xb2121723743c9402bd88c5dc59f309ac79307c9a

owner has several privileges over contract, all of them are discussed in the report

current owner of Dvidend Tracker is:

0xa5f8128d04692656a60f17b349c2146c48e6863a

which is the Fight4Hope token contract itself

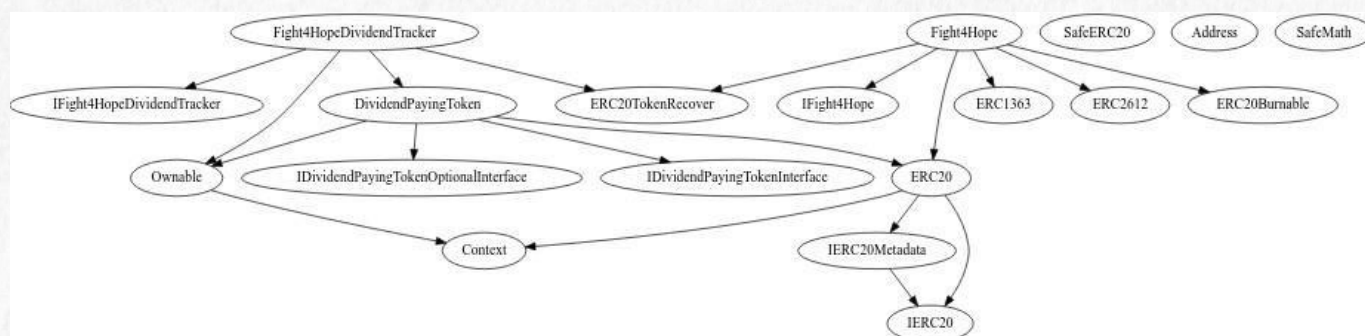
Audit Scope:

all of below contracts are in this audit scope

[Fight4Hope.sol](#)

[Fight4HopeDividendTracker.sol](#)

Inheritance Tree:



AUDIT SUMMARY

Summary

- **Fight4Hope** is implementing a bridge functionality, bridge contracts are not in this auditscope, developer of the contract is well aware of all best bridge security practices.
- **Fight4Hope** gives **RBA** tokens as a reflection to holders, this **RBA** token has 20% tax on buy and 20% tax on sell | [Chart](#)
- **Fight4Hope** uses a dividend tracker contract to distribute rewards between holders
- **Fight4Hope** is implementing [EIP-2612](#) which is ERC20 + additional functionalities to support transferring tokens with an off-chain provided signature and without the need for a token holder to approve its tokens.

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 Summary

High Risk Findings: 3

Medium Risk Findings: 1

Low Risk Findings: 0

Suggestions & discussion: 3

Gas Optimizations : 2

Fight4HopeDividendTracker is a popular contract used by many tokens, we didn't spend much time testing its functionality, we just read the code to make sure there is no changes that need to be audited. one important thing is that reward token, or dividend token can't be zero address, which is not in this case and cannot be changed also.

Another consideration is that the dev must not be able to withdraw reward tokens from dividend contract, which is not able to do so.

High Risk Findings

Fight4Hope team acknowledged all these major issues and they are aware of them.

Centralization - Bridge Vault is an **EOA** address as of now, any access to this wallet's private key can damage token price badly if used in a malicious way.

Centralization - Owner is able to **disable** transferring

```
function setTransfersEnabled(bool enabled) external override onlyOwner {
    require(transfersEnabled != enabled, "Already set to this value");
    transfersEnabled = enabled;
    emit LogSetTransfersEnabled(enabled);
}
```

only whitelisted wallets are able to bypass this limit in **_transfer** function:


```
if (!transfersEnabled) {  
    require(whitelistedAddresses[from], "Fight4Hope: Transferring is disabled");  
}
```

Recommendation:

give clear explanations about this function or remove it

Centralization - Owner is able to disable dex (Pancake Swap) trades

```
function setDexTradesEnabled(bool enabled) external onlyOwner {  
    require(dexTradesEnabled != enabled, "Already set to this value");  
    dexTradesEnabled = enabled;  
    emit LogSetDexTradesEnabled(enabled);  
}
```

only whitelisted wallets are able to bypass this limit

```
if(!dexTradesEnabled && (automatedMarketMakerPairs[to] ||  
automatedMarketMakerPairs[from])){  
    require(whitelistedAddresses[from], "Fight4Hope: dex trades are disabled");  
}
```

Recommendation:

give clear explanations about this function or remove it

Centralization - Owner is able to set **buy** & **sell** taxes each up to 50%

```
function updateBuyFees (
    uint256 _dividendFee,
    uint256 _liquidityFee,
    uint256 _marketingFee
) external override onlyOwner {
    buyDividendFee = _dividendFee;
    buyLiquidityFee = _liquidityFee;
    buyMarketingFee = _marketingFee;
    buyTotalFees = buyDividendFee + buyLiquidityFee + buyMarketingFee;
    require(buyTotalFees <= 5000, "Max fee is 50%");
    emit LogUpdateBuyFees(buyDividendFee, buyLiquidityFee,
buyMarketingFee);
}
```

Recommendation:

set a reasonable amount for tax limits

Logical

Functions lock and unlock are used to transfer tokens from and to bridge vault, since approval is needed for bridge to be able to spent this wallet tokens, then owner must have access to bridge vault private key and this can open new centralization issues.

Medium Risk Findings

Centralization - Owner is able to withdraw LP tokens using **recoverERC20**, this LP tokens are the tokens that contract receives in exchange for adding a portion of collected taxes as liquidity

```
defaultDexRouter.addLiquidityETH{value: ethAmount}((
    ...
    address(this),
    ...
);
```


Gas Optimizations

Too much use of **balanceOf** function for same values over and over again (at `_transfer` function), define a memory variable to lower trades gas amount

instead assign a memory variable to them to lower buy/sell gas

Define **_startSupply** and **maxWalletToken** as constant, these values never change inside the contract

Redundant check, transaction will revert if this condition is not met

```
uint256 senderBalance = balanceOf(from);  
require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
```

Suggestions

Delete override for **state variables**, there are many state variables with override keyword, these variables are not implemented in any of parent contracts

Change **contractBalanceReceipient** to **recepientBalance** (`_transfer` function)

```
uint256 contractBalanceReceipient = balanceOf(to);  
require(  
    contractBalanceReceipient + amount <= maxWalletToken,  
    "Fight4Hope: Exceeds maximum wallet token amount."  
);
```

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.