expelee

Building the Futuristic Blockchain Ecosystem

Audit Report

FOR

Forever Farm

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	Done
Audit Date	21 Oct 2022

-Team Expelee

PROJECT DESCRIPTION

Forever Farm

Forever Farm is an innovative defi protocol with a selfsustaining, never ending yield farm!

FOREVER FARM

SELF SUSTAINING YIELD FARMING

expelee

expelee.com

Social Media Profiles

Forever Farm

https://foreverfarm.pro/

https://t.me/ForeverFarm

https://twitter.com/ForeverFarmBSC

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

-Team Expelee

CONTRACT DETAILS

Token Name
Forever Farm

Symbol
F(infinity)F

Network
BSC

Language
Solidity

Contract Address (Verified)
0x51d0a28B1275014FF85C68A991862f7F30b9710b

Token Type
ERC 20

Decimals
18

Compiler
-

Total Supply
100,000

Contract SHA-256 Checksum:
0bcd5cc7aac93dec7c895531092b3ed6a2d1599298402da4013b81e5178d2112

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

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.

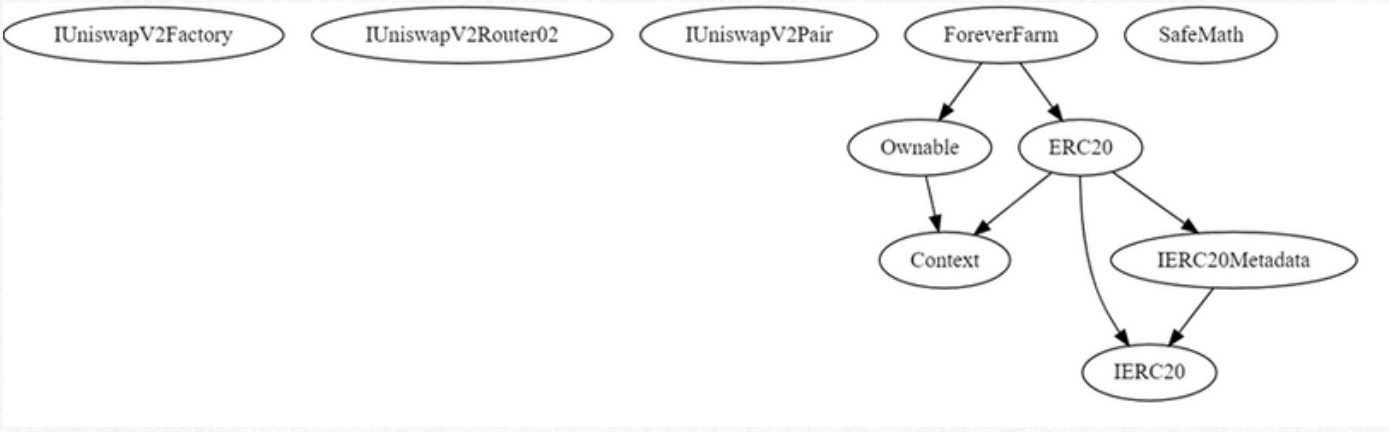
AUDIT SUMMARY

Ownership:

No ownership, not deployed yet.

Contracts & Inheritance Tree:

ForeverFarm token is inheriting from this contracts



Summary

- Auto Burn Mechanism up to 9% every 3 minutes
- Manual Burn Mechanism up to 10% every 10 minutes
- Taxes are going into marketing and fund wallets
- Anti-sniper mechanism, this anti-sniper should be disabled by owner manually

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 followingkey 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:11
- Medium Risk Findings:0
- Low Risk Findings:0
- Suggestions & discussion: 2
- Gas Optimizations : 3

High Risk Findings

Centralization – Currently, LP tokens received from auto-liquidity mechanism are added to the contract, a malicious owner is able to use "removeToken" to remove this LP tokens and use them to remove a portion of liquidity.

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        address(lpLocation),
        block.timestamp
);
```

Suggestion:
send LP tokens to dead address

Centralization – Owner is able to change "lpLocation" wallet to an arbitrary address. lpLocation is a wallet address that receives LP tokens generated from auto-liquidity mechanism. a malicious owner is able to use this tokens to remove a portion of liquidity pool tokens.

```
function updateLPLocation(address _address) public onlyOwner {
    lpLocation = _address;
}
```

Suggestion:
send LP tokens to dead addres

Centralization – owner is able to set multiple arbitrary addresses as snipers, snipers are not able to sell/transfer tokens.

```
function addBulkSniper(address[] memory _list) public onlyOwner{
    for(uint256 i=0;i<_list.length;i++){
        sniper[_list[i]] = true;
    }
}
```

Suggestion:
give explanations about usage of this function or remove it.

Centralization – owner is able to disable trading at any time, by disabling trades, no one is able to buy/sell/transfer tokens except whitelisted wallets.

```
function updateTradingEnabled(bool enabled) public onlyOwner{
    tradingActive = enabled;
}
```

Suggestion:
make sure that you are not able to disable trading after enabling it.

Centralization – owner is able to set buy / sell fees each one up to 99%.

```
function updateBuyFees(uint256 _marketingFee, uint256 _liquidityFee, uint256
_devFee) external onlyOwner {
    buyMarketingFee = _marketingFee;
    buyLiquidityFee = _liquidityFee;
    buyDevFee = _devFee;
    buyTotalFees = buyMarketingFee + buyLiquidityFee + buyDevFee;
    require(buyTotalFees <= 99, "Must keep fees at 99% or less");
}
```

Suggestion:
set a reasonable limit for max buy/sell taxes.

Logical - at updateMaxWalletAmount function, require statement is not matching returned error message:
require(newNum >= (totalSupply() * 5 / 1000)/1e18, "Cannot set maxWallet lower than 0.5%");
because denominator at right side of the statement is also multiplied by 1e18, this means that we can set newNum to 500 (not in wei), which is 0.00000000000000000005% of total supply.

Suggestion:
remove "/1e18" from the require statement

Centralization – Every buyer gets blacklisted if "noSniper" is set to "true" owner must call "gotcha" function two times in order to disable anti-bot mechanism, any unexpected issue or delay in disabling anti-sniper can blacklist a large number of trades

Suggestions:
remove "/1e18" from the require statement.

Centralization – owner is able to to transfer up to 9% of pool tokens to "fund" wallet (a wallet which is changeable by owner and is currently set to owner's wallet) using "manualBurnLiquidityPairTokens" function.

```
if(tradingActive && noSniping){
   sniper[to] = true;
}
```

Suggestions:
its not clear what is usage of this function as its name is not matching what its actually doing. we suggest to send this tokens to dead address or reduce them from total supply

Centralization - as of now 0.25% of Liquidity pool tokens go into owner's wallet(fund wallet) every 3600 seconds(10 minutes), this operation is done by "autoBurnLiquidityPairTokens" function.

```
if (amountToBurn > 0){
    super._transfer(uniswapV2Pair, fund, amountToBurn);
}
```

Suggestion:
its not clear what is usage of this function as its name is not matching what its actually doing. we suggest to send this tokens to dead address or reduce them from total supply

Centralization –Owner is able to change percentage and ratio of auto burnings, curerntly 0.25% of liquidity pool tokens is getting burned (not realy burned, refer to last issue) every 10 minutes, owner is able to change this percentage up to 10%. (manualBurnLiquidityPairTokens function)

```
if (amountToBurn > 0){
    super._transfer(uniswapV2Pair, fund, amountToBurn);
}
```

Suggestion:

its not clear what is usage of this function as its name is not matching what its actually doing. we suggest to send this tokens to dead address or reduce them from total supply

Logical – Sells from non-sniper wallets or owner wallet or any other wallet that has tokens but is not recognized as sniper can blacklist liquidity pool if anti-sniper is on.

```
if(tradingActive && noSniping){
    sniper[to] = true;
}
```

Suggestion:

make sure that "to" is not liquidity pool

Gas Optimizations

- redundant check at _transfer: to != address(0) &&
- redundant checks at _transfer below transferDelayEnabled check: to != owner() && to != address(uniswapV2Router), _to will never be router and we already checked to != owner() in previous if statement

Suggestions

- you can use up to 3 indexed keywords in events for fixed size datas.
- do not enable trading before adding liquidity, doing this can blacklist(add to snipers list) liquidity pool

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.