Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

SECFAIL

TOKEN OVERVIEW

Risk Findings

Severity	Found
🔴 High	2
🟠 Medium	0
🟡 Low	1
🔵 Informational	1

Centralization Risks

Owner Privileges	Description
🟢 Can Owner Set Taxes >25% ?	Not Detected
🟢 Owner needs to enable trading ?	Not Detected
🔴 Can Owner Disable Trades ?	Detected
🟢 Can Owner Mint ?	Not Detected
🔴 Can Owner Blacklist ?	Detected
🟢 Can Owner set Max Wallet amount ?	Not Detected
🟢 Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

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	Failed
KYC Verification	-
Audit Date	21 June 2023

CONTRACT DETAILS

Token Name: SecFail

Symbol: SECFAIL

Network: Binance smart chain

Language: Solidity

Contract Address:
0x507bc9e7e64c341837BaE6650C1856F3cDA46f3e

Total Supply: 100,000,000

Owner's Wallet:
0xba549a7A9404Ed3Bab294E4a38aD7627e9e8B268

Deployer's Wallet:
0xba549a7A9404Ed3Bab294E4a38aD7627e9e8B268

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

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps 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 Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces 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.

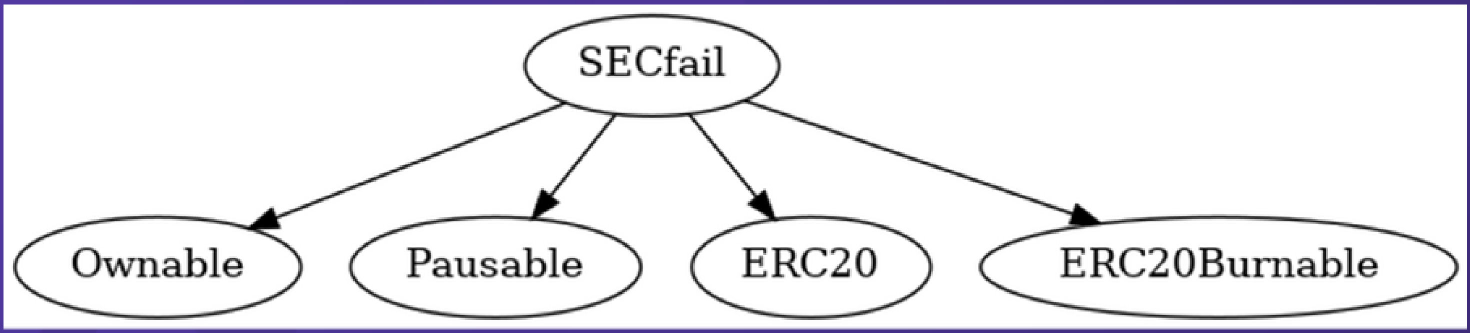
Low Risk

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

Informational

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

INHERITANCE TREES



$$\begin{tikzcd}
 & & \text{SECfail} \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
\text{Ownable} & \text{Pausable} & & \text{ERC20} & \text{ERC20Burnable}
\end{tikzcd}$$

TESTNET VERSION

Adding Liquidity ✅
Tx:
https://testnet.bscscan.com/tx/0x2c83e220500b415779dd9aa89a726bbe827604dd9685f730b93e22a5b468429f

==

Buying ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x622397b309e9534c2e4f41c27c6bc055a484b32cdf34d0eb4aa4fea353f84614

==

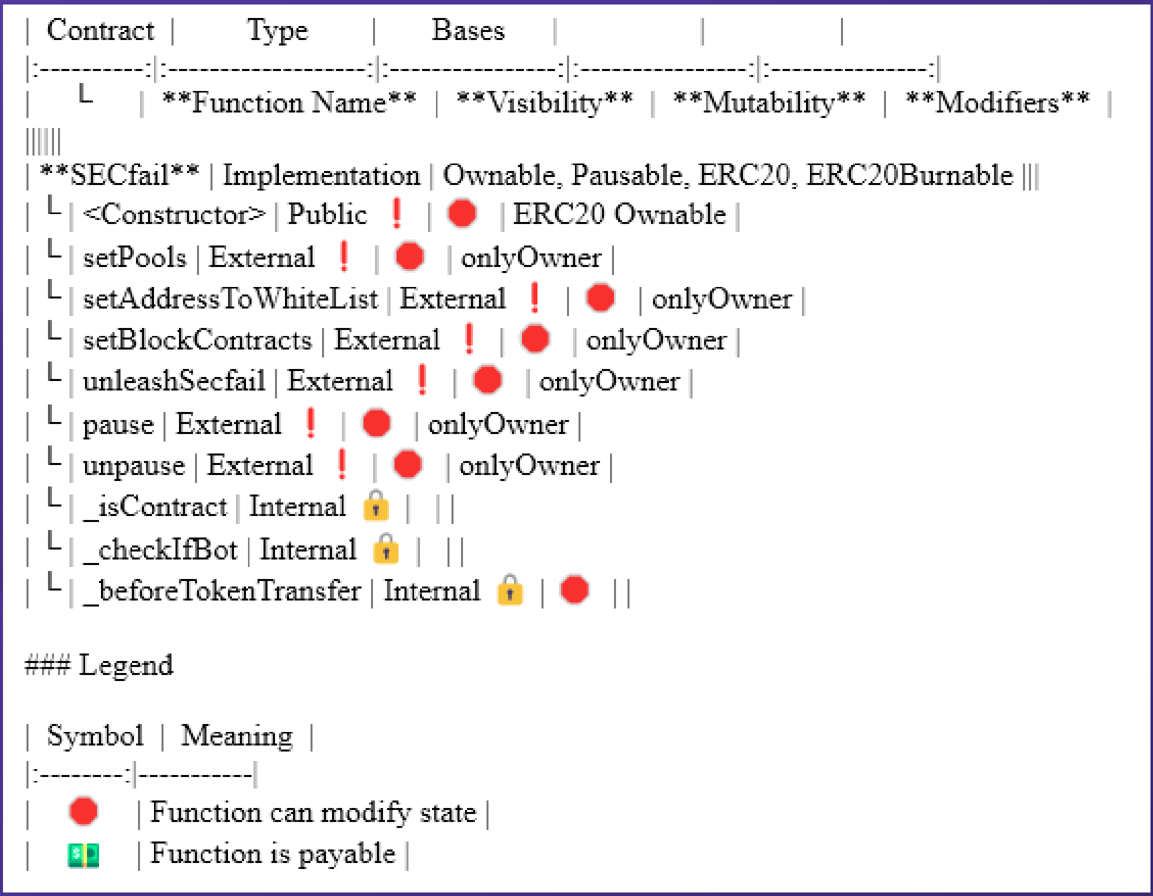
Selling ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0xbfcb24269f702dd5b6d052cc67624090747b6ef5524530c440330bea89dc3dfa

==

Transferring ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x596f69887810a61d30b037db3610922847ef5ed76ef4dbbc81e5cd050869d740

==

FUNCTION DETAILS



```
| Contract |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |  **Modifiers**  |
||||||
| **SECfail** | Implementation | Ownable, Pausable, ERC20, ERC20Burnable |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 Ownable |
| └ | setPools | External ❗ | 🛑  | onlyOwner |
| └ | setAddressToWhiteList | External ❗ | 🛑  | onlyOwner |
| └ | setBlockContracts | External ❗ | 🛑  | onlyOwner |
| └ | unleashSecfail | External ❗ | 🛑  | onlyOwner |
| └ | pause | External ❗ | 🛑  | onlyOwner |
| └ | unpause | External ❗ | 🛑  | onlyOwner |
| └ | _isContract | Internal 🔒 |   | |
| └ | _checkIfBot | Internal 🔒 |   | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |


### Legend

|  Symbol  |  Meaning  |
|:--------:|-----------|
|    🛑    | Function can modify state |
|    💵    | Function is payable |
```

MANUAL REVIEW

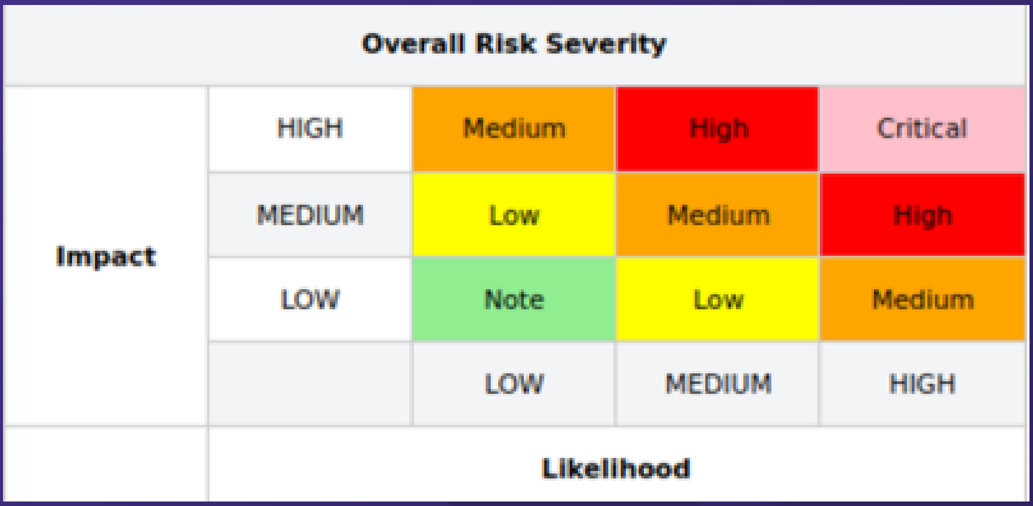
Severity Criteria

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

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
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			

HIGH RISK FINDING

Category: Centralization
Subject: Pausing/Unpausing buy/sell/transfer
Status: Open
Severity : High

Overview

Owner is able to pause/unpause contract. When contract is paused, only whitelisted wallets are able to buy/sell/transfer.

```
  function pause() external onlyOwner {
    _pause();
 }

 function unpause() external onlyOwner {
    deadblockStart = block.number;
    _unpause();
 }

 function _beforeTokenTransfer(address sender, address recipient, uint256
amount) internal override {
    //rest of the code
    if (paused() && !whitelist[sender]) {
      revert ContractPaused();
    }
    //rest of the code
 }
```

Suggestion

There are multiple ways to solve this issue:
- Unpause trades and renounce ownreship
- Transfer ownership of the contract to a trusted 3rd party (e.g. pinksale safu developer)

HIGH RISK FINDING

Category: Configuration
Subject: Hidden blacklist
Status: Open
Severity : High

Overview
A malicious owner is able to set an arbitrary address as a “pair” in the contract (using setPools function). In this situation if _blockContract is enabled and if address of liquidity pool is not whietlisted, seller wont be able to complete the transaction.

```
if (isBuy) {
    // in this case, recipient is address of liquidity pool in a sell transaction
  if (_blockContracts && _checkIfBot(recipient)) {
    revert NotAllowed();
  }

//if liquidity pool is not whietlisted, this function will return “true’
 function _checkIfBot(address _address) internal view returns (bool) {
   return (block.number < DEADBLOCK_COUNT + deadblockStart ||
_isContract(_address)) && !whitelist[_address];
 }
```

Suggestion:
Ensure that a non-contract address can’t be set as a valid pool.

```
function setPools(address[] calldata _val) external onlyOwner {
  for (uint256 i = 0; i < _val.length; i++) {
    require(isContract(_val[i]), “address must be contract’);
    address _pool = _val[i];
    poolList[_pool] = true;
    emit LiquidityPoolSet(address(_pool));
  }
}
```

LOW RISK FINDING

Category: Configuration
Subject: Anti-bot can be reset
Status: Open
Severity : Low

Overview

unpause function sets deadblockStart (starting block of the trades) to current block.number regardless of whether this value was set before or not. This means anti-bot will be enabled for 3 blocks each time after calling unpause

```
    function _checkIfBot(address _address) internal view returns (bool) {
      return (block.number < DEADBLOCK_COUNT + deadblockStart ||
_isContract(_address)) && !whitelist[_address];
  }
```

Suggestion

Ensure that deadblockStart can only be update one time

```
  function unpause() external onlyOwner {
    if(deadblockStart == 0){
      deadblockStart = block.number;
    }
    _unpause();
  }
```

INFORMATIONAL

Category: : Lost funds
Subject: ERC20 token can’t be withdrawn
Status: Open
Severity : Informational

Overview
There are no function in the contract to be able to withdraw Stuck ETH or ERC20 tokens from the contract .

Suggestion
Implement a function for withdrawing stuck ERC20 tokens from the contract

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project 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. Alway 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.

expelee

Building the Futuristic Blockchain Ecosystem