expelee

Building the Futuristic Blockchain Ecosystem

Security Audit Report

FOR

Fundex Exchange

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	Passed
Audit Date	24 Feb 2023

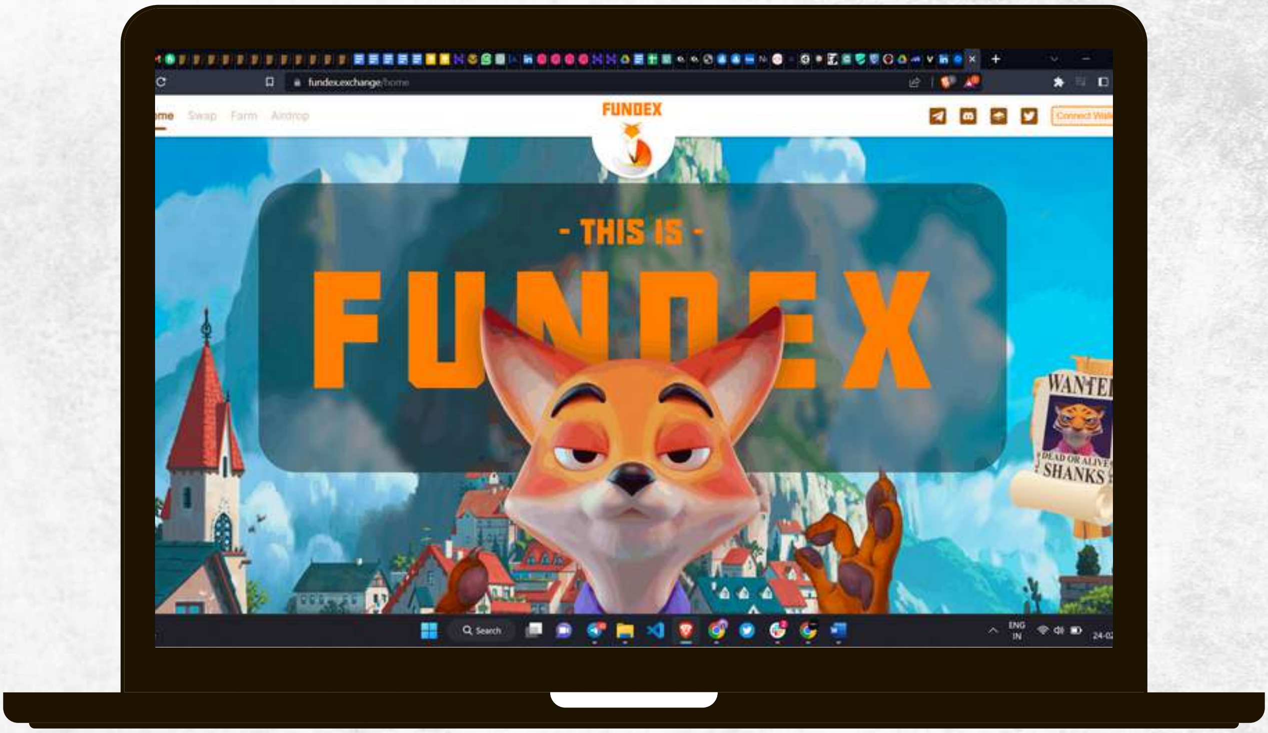
PROJECT DESCRIPTION

Fundex Exchange

Governance: FUN token holders can participate and vote on governance decisions.
Liquidity Incentives: Users can earn Fun tokens as rewards by providing liquidity with stablecoins to the liquidity pool.
Boosting: Users can earn additional Fun from the Boosting Pool by locking Fun tokens. The Boosting Pool incorporates voting escrow Fun (veFun) for rewards accrual. Locking any amount of Fun applies a boost to all stablecoin pools.

Social Media Profiles

Fundex Exchange



https://fundex.exchange/home

https://t.me/Fundexexchange

https://twitter.com/Fundexexchange

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

-Team Expelee

CONTRACT DETAILS

Token Name

Fun DAI Asset

Symbol

LP-DAI

Network

BSC

Language

Solidity

Contract Address (Verified)

0x7b5B4c5e381c7Dc116BF9dA8dDd1c96a3bd6C1cb

Token Type

ERC20

Total Supply

0 initialy

Contract SHA-256 Checksum:

f1a8b392a6f39e4f04764f110893db2f2fc36193

Owner Wallet

0xD8693368d37b502eD54C315e38efDE7EB3dCDB5c

Deployer Wallet

0xD8693368d37b502eD54C315e38efDE7EB3dCDB5c

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	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

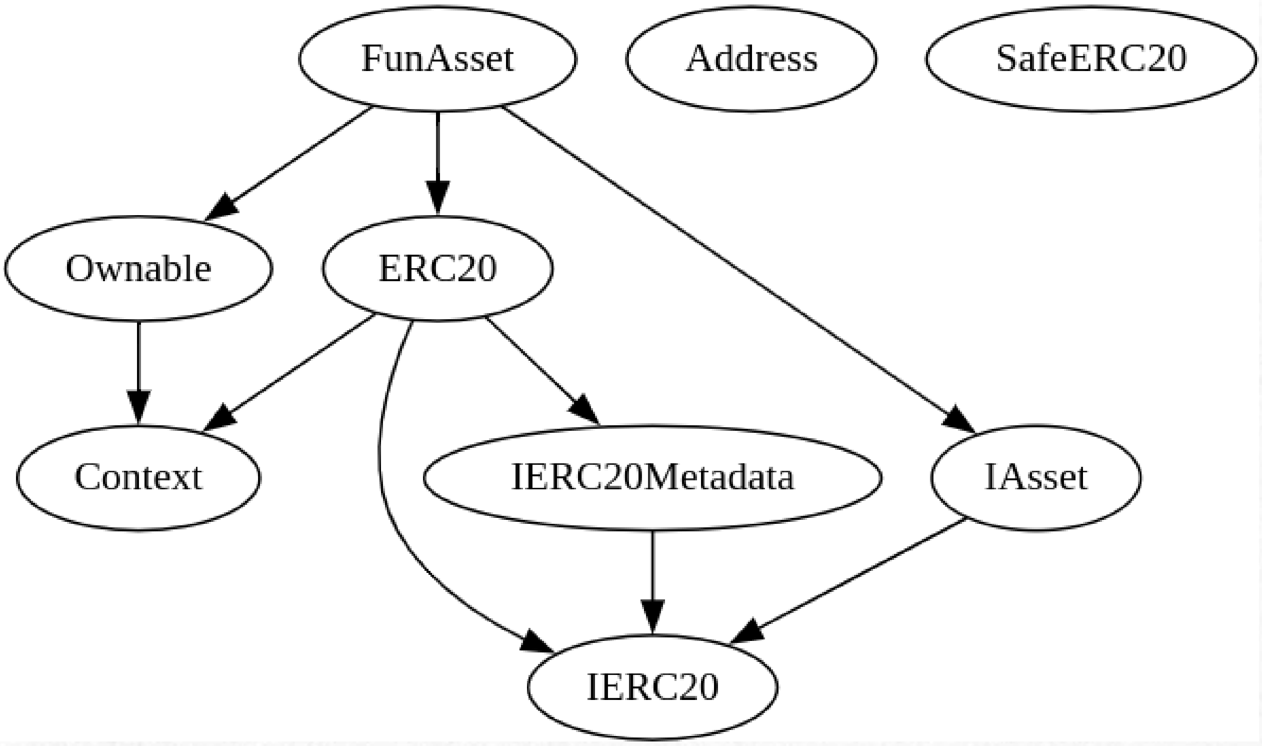
Used Tools:

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 mint new tokens
- Owner is not able to set buy/sell/transfer taxes (0% static)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades

Functional Tests

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

minted 10,000 tokens

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x7a00a31cb4c1d9635469d65a0eb76cc1013d3b6c00fde6d9ffc29be240c20cd5

2- Buying (0% tax) (passed):
https://testnet.bscscan.com/tx/0xdd888025f3e3c94a7acf123e008e098a279a0c56dc83b9035f5beef533fba677

3- Selling (0% tax) (passed):
https://testnet.bscscan.com/tx/0x1d0b9958d92ba05f5e79d77323c58ac209f96f75d48c0fa5f46e349c9f7309cb

4- Transferring (0% tax) (passed):
https://testnet.bscscan.com/tx/0xbe0dad30fa0fddeaf23100c43ad68f0b1d5725ce9aac2257cde0a93d3edca7bb

MANUAL AUDIT

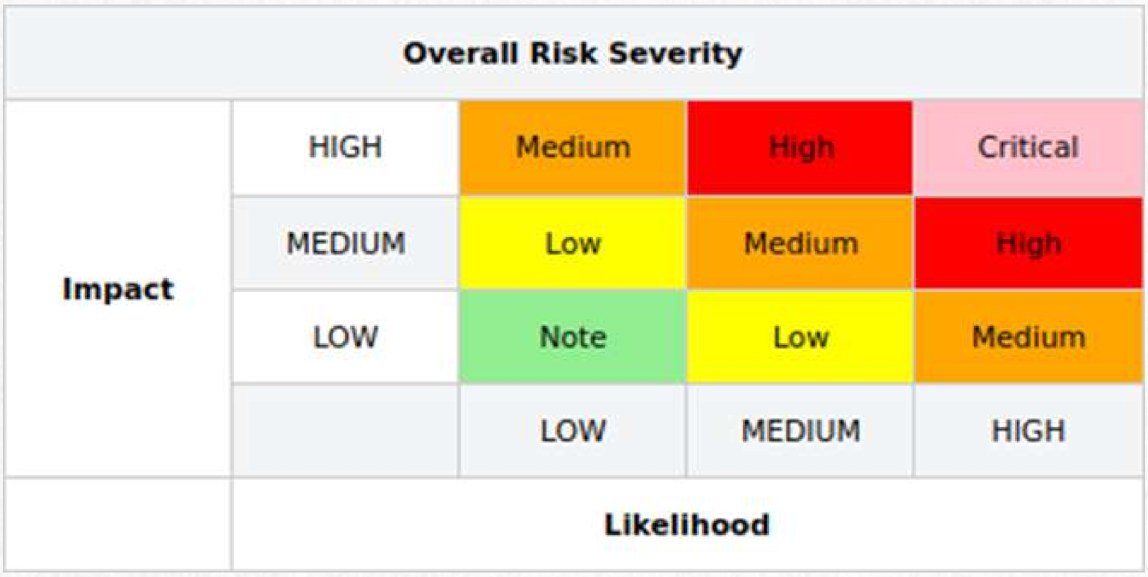
Severity Criteria

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

- High Risk Findings:2
- Medium Risk Findings:0
- Low Risk Findings:0
- Suggestions & discussion: 0
- Gas Optimizations : 0

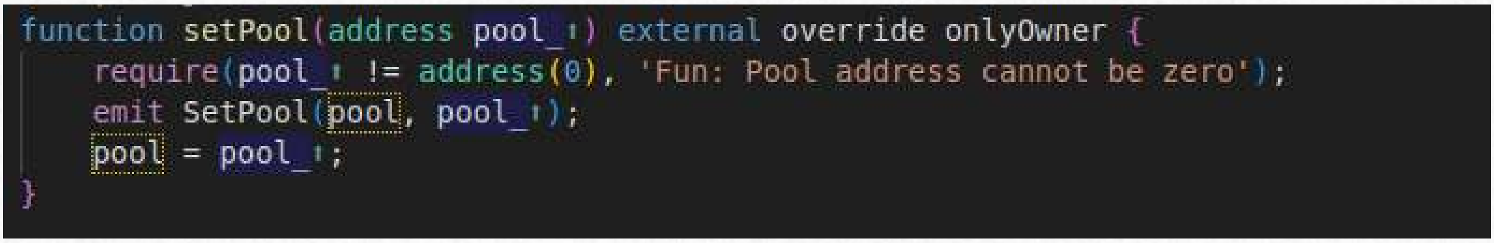
High Risk Findings

The contract includes mint and burn functions that are controlled by the liquidity pool contract, but the pool contract itself is controlled by an unknown owner, and we do not yet have information about access control or how the contracts are deployed and initialized. While it is not uncommon for liquidity pool contracts to include these functions, the lack of clarity around ownership and access control presents a potential security risk that must be investigated further. **These risks are mentioned below:**

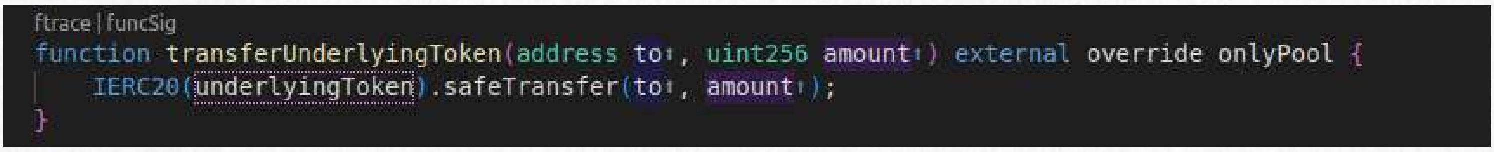
Centralization – delegation of pool actions:

the contract allows owner to change the pool address to an arbitrary address, which enables new pool address to perform **pool actions** like withdrawing underlying token or burning tokens from any wallet.

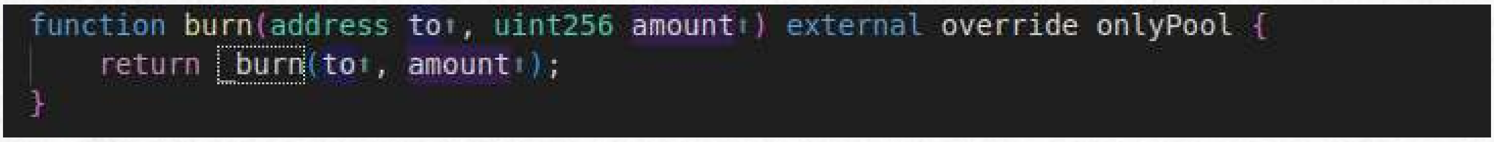
The contract contains a **'transferUnderlyingToken'** function that can be called by the pool address to transfer the underlying token, and a **'setPool'** function that allows the owner to modify the pool address.



```
function setPool(address pool_) external override onlyOwner {
    require(pool_ != address(0), 'Fun: Pool address cannot be zero');
    emit SetPool(pool, pool_);
    pool = pool_;
}
```



```
ftrace | funcSig
function transferUnderlyingToken(address to, uint256 amount) external override onlyPool {
    IERC20(underlyingToken).safeTransfer(to, amount);
}
```

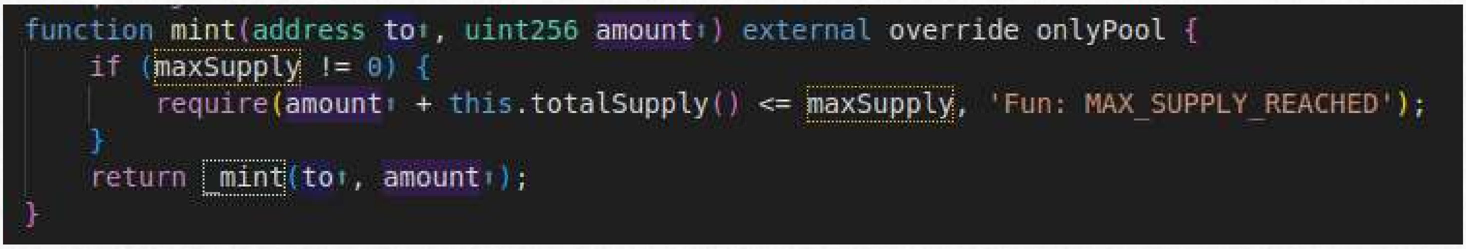


```
function burn(address to, uint256 amount) external override onlyPool {
    return _burn(to, amount);
}
```

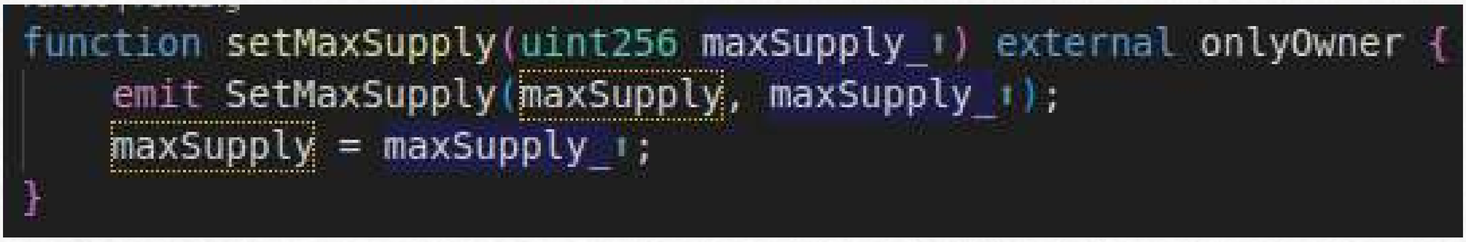
High Risk Findings

Centralization – Minting:

we have found a centralization risk in the contract that allows the “pool” to mint unlimited tokens and change the max supply limit, which is intended to restrict the total number of tokens that can be minted. This could result in uncontrolled inflation of the token's supply and potentially diminish its value. The smart contract includes a 'mint' function that is only accessible to the owner and a 'setMaxSupply' function that allows the owner to modify the max supply limit. While the 'mint' function does include a check for the max supply limit, this can still be circumvented by the owner by modifying the value of the max supply limit using the 'setMaxSupply' function.



```
function mint(address to, uint256 amount) external override onlyPool {
    if (maxSupply != 0) {
        require(amount + this.totalSupply() <= maxSupply, 'Fun: MAX_SUPPLY_REACHED');
    }
    return _mint(to, amount);
}
```



```
function setMaxSupply(uint256 maxSupply_) external onlyOwner {
    emit SetMaxSupply(maxSupply, maxSupply_);
    maxSupply = maxSupply_;
}
```

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.