expelee

Building the Futuristic Blockchain Ecosystem

Security Audit Report

FOR

8Bit Prodex Core Contracts

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
KYC Verification	Done
Audit Date	16 Jan 2023

Audit Passed With no Risk

-Team Expelee

PROJECT DESCRIPTION

8BitEARN

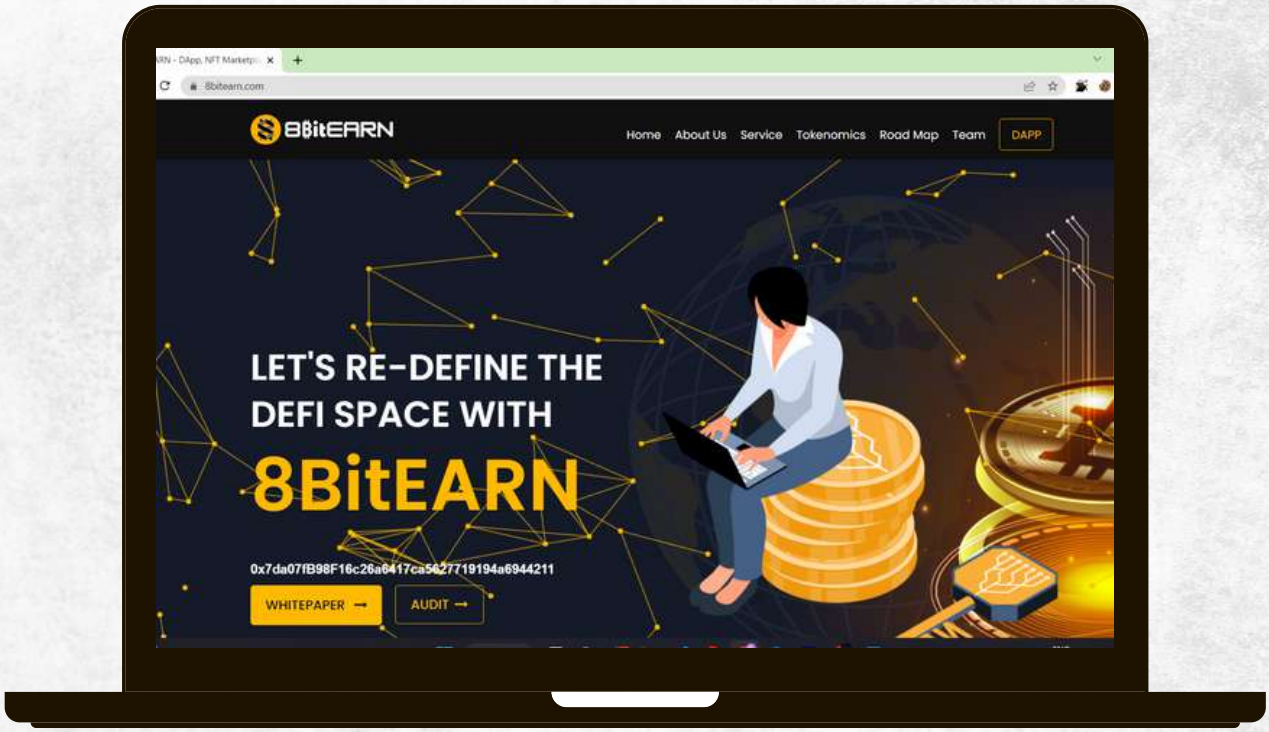
$8Bit is a hyper-deflationary BEP-20 native token of the 8BitEARN Ecosystem, that opens numerous passive income streams & benefits to holders by offering BTC Reflection, Staking Rewards, Monthly Diamond Hand Rewards, Quarterly Revenue Distribution, BUSD Credit Facility, Investment Insurance at ProPAD, Farming & Cashback at ProDEX, several DeFi benefits to 8Bit NFT holders with flawless NFT trading at own NFT Marketplace, transparency & integrity through DAO Governance and many more.

expelee

expelee.com

Social Media Profiles

8BitEARN



https://www.8bitearn.com/

https://t.me/official8BitEARN

https://twitter.com/8BitEARN

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

-Team Expelee

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

Audit Scope:

The **Expelee** audit team carried out a general code review and security analysis of the smart contracts of Prodex-v2-core, this audit focuses on core contracts of **ProDex-V2** which are:

ProDex Factory:

Responsible for creating new pools, deployed and verified at:
https://bscscan.com/address/0x12B99aa33A6E8995AC78bB033d95FBD5C7C17a61#code

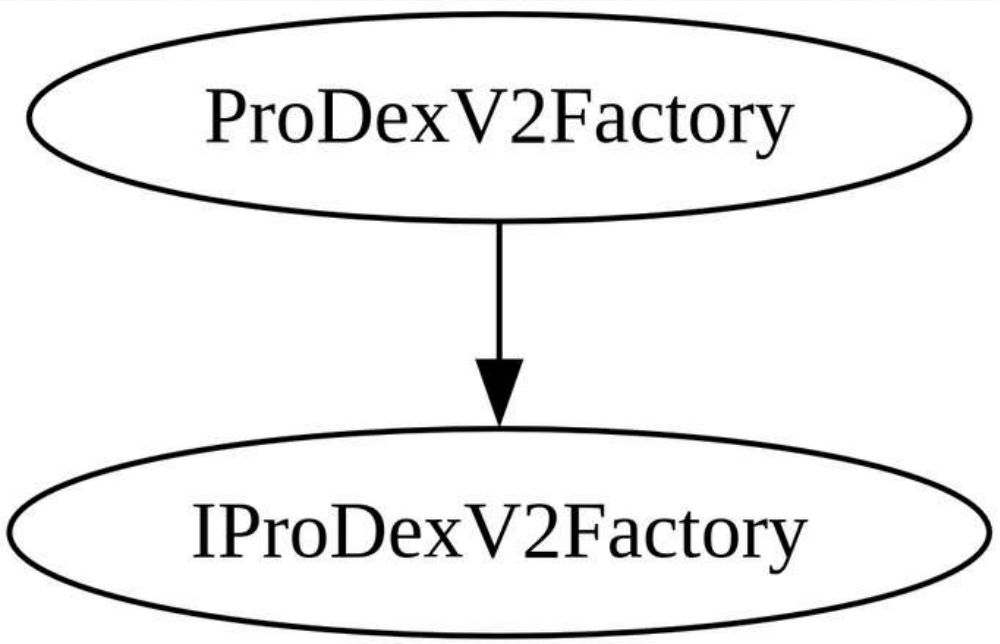
---

ProDex Pools:

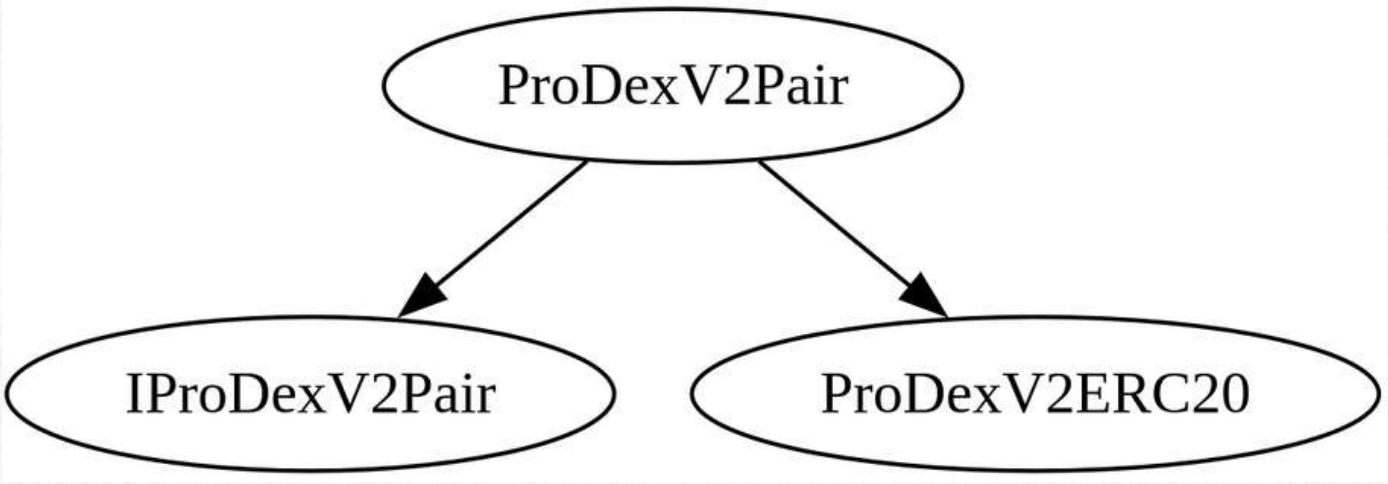
For handling swaps and liquidity.
represented in Factory contract at above address

Inheritance Trees:

Prodex factory:



ProDex pair, which is also an standard ERC20 token and is used as liquidity pool token (LP token):



Features

Creating pairs:

Prodex factory uses **create2** opcode to deploy new pool contracts, create2 opcode is used to calculate contract address before actually being deployed to the chain. This opcode is also used in router libraries to reduce overall gas fees.
After creating a pair, it will be registered in the factory contract.

initial liquidity can only be provided by Prodex owners or whitelisted wallets, but there is no restrictions to provide or remove liquidity after that.

Pairs:

Pairs can be created between 2 arbitrary ERC20 tokens.

Swapping:
- pool rates, inputs and outputs:

prodex uses famous **constant product formula** to maintain liquidity pool ratio:
x * y = k
input or output amounts are calculated in a way to **maintain constant k. this also prevents pools from being fully drained by applying up to max 50% price impact on trades:**

output = (output * input_reserves) / (output + output reserves)

- Swap fees:

prodex applies 0.03% swap fee on each trade, this fee is accumulated and given to liquidity providers, Prodex doesn't charge protocol fees as of now

Providing Liquidity:

initial liquidity can only be provided by Prodex owners or whitelisted wallets, but there is no restrictions to provide or remove liquidity after that.

Pool Shares:

initial pool shares are calculated by geometric mean of initial reserves:
Math.sqrt(reserves0 * reserves1)
this way, pool supply has a geometric connection to current pool reserves. pool share percents after initial liquidity, are calculated in this way:

T1 = first pair
T2 = second pair
T1R = first pair reserves
T2R = second pair reserves
TPS = total pool shares
pool shares to get = minimum(T1 * TPS / T1R, T2 * TPS/T2R)

this means providing liquidity in a wrong ratio than pool ratio can result in losing tokens (minimum amount of pool share is considered) This is to punish bad liquidity providing.
LP tokens are sent to liquidity provider and Prodex owner do not have any controls over other liquidity provider tokens

Removing Liquidity:

removing liquidity cannot be restricted by ProDex owners for liquidity providers
by removing liquidity, backed tokens are sent to liquidity provider, backing tokens are calculated in this way:

PSB = Pool share balance
TPS = total pool shares
T1R = first pair reserves
T2R = second pair reserves

T1 to get = PSB * T1R / TPS
T2 to get = PSB * T2R / TPS

Price Oracles:

The Prodex-V2 protocol introduces the time-weighted average price accumulators price0CumulativeLast, price1CumulativeLast. The accumulators track the Pair cumulative time-price at the end of each block. Sampling the accumulator at two points in time, taking the difference, and dividing by the elapsed time yields the time-weighted average price of the price in that Pair at the ends of the blocks during that time interval. From a robustness perspective, this differs from the instantaneous price in two crucial ways:

• The averaged price depends on prices that appeared in the past, proportionally to how long they appeared for, and the oracle consumer can choose the length of the period for averaging.

• The averaged price is not influenced by prices that appeared within a block, but only by the final price at the end of a block. In particular, the average is not affected by prices arising during synchronous execution of multiple trades within a block.

The first point means that in order to manipulate an oracle which uses a longer averaging period, an attacker would need to maintain a manipulated price for a longer period of time. The second means that an attacker must maintain the manipulated price at the end of a block in order to have an effect on the average, which is expected to maximize the attacker's cost by reducing the likelihood that they will be the one to "de-manipulate" the price.

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

ProdexV2 implemented latest Dex design and practices used in many approved and secure Dexes such as Uniswap V2 & Pancakeswap V2 etc. there was no issues found in the contarcts other than some suggestions which.

- Centralization: 1
- Logical: 0
- Suggestions: 2

Centralization:

Risk Level: Low

Only Prodex operators are able to create a pair, this means no one other than prodex operator and whitelisted wallets are able to create a pair (initial liquidity)

Suggestions

Factory: replace allPairs array with a counter

The allPairs array in the Factory is primarily used to maintain a count of all created Pairs and the audit team noted that it could be replaced by a simple counter. The Prodex team decided to keep the array implementation in case iteration over the array might be required. They were also advised that any code which iterates over the allPairs array should paginate, or otherwise break up the access into constant-gas chunks, and moreover that iterating over the array from an external contract would be very costly, requiring one call per element read.

Replace math libraries with an inherited contract

The ProdexV2Pair contract depends on three libraries, UQ112x112, SafeMath and Math, providing additional internal functions and custom syntax for uint256 safemath. Including these internal methods in the ProdexV2Pair directly (or inheriting them from another contract) instead makes creating new pairs about 15000 gas cheaper.

Resolved: Since this change would make the use of custom syntax for uint256 safemath impossible, its decided to not implement this suggestion.

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.