VitalBlock

FOR

# LUXPAD TOKEN

WEBSITE

https://www.luxpad.io

TELEGRAM

https://t.me/luxpad_official

# SMART CONTRACT AUDIT

Twitter:
twitter.com/vital_blockA

Telegram:
t.me/vital_Block

# Disclaimer

Vital Block Solidity reports are not, nor should be considered, an "endorsement" or "disapproval" of any project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Vital Block to perform a security review.

Vital Black Solidity Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analysed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

Vital Block Solidity Reports should not be used in any way to make decisions around investment or involvement with any project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort. Vital Block Solidity Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Vital Block Solidity's position is that each company and individual are responsible for their own due diligence and continuous security. Vital Block Solidity's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyse

## What is a Vital Block Audit report?

•A document describing in detail an in-depth analysis of a particular piece(s) of source code provided to Vital Block Solidity by a Client.

•An organized collection of testing results, analysis and inferences made about the structure, implementation, and overall best practices of a particular piece of source code.

•Representation that a Client of Vital Block Solidity has indeed completed a round of auditing with the intention to increase the quality of the company/ product's IT infrastructure and or source code.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | **LUXPAD TOKEN** |
| Description | **Luxpad is a decentralized trustless launchpad platform where you can create your ERC-20 and BEP-20 token, NFT smart contract, fair-launch, or presale launchpad without any knowledge of coding using the Luxpad platform.** |
| Platform | **Binance Mainnet** |
| Mainnet Contracts: | **0xFde55fA832dA95D0c45240F0450fd86Caf2E9Ac1** *LUXPAD TOKEN* (LXPD) |
| Files: | LuxpadToken.sol |

## Audit Summary

| | |
|---|---|
| Delivery Date | **July August 12 2022** |
| Method of Audit | **Security Static Analysis** |
| Timeline | **Story Points 100** |

## Vulnerability Summary

| | |
|---|---|
| Total Issues Found | **1** |
| Total Issues Resolved | **1** |
| Total Critical | **0** |
| Total High | **1** |
| Total Medium | **2** |
| Total Low | **0** |
| Total Informational | **2** |

# Executive Summary

## Our Audit Methodology

- **STEP 1**
  A manual line-by-line code review to ensure the logic behind each function is safe and secured against common attack vectors.

- **STEP 2**
  Simulation of hundreds of thousands of Smart Contract Interactions on a test and Mainnet blockchain using a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

- **STEP 3**
  Consultation with the project team on the audit report pre-publication to implement recommendations and resolve any outstanding issues.

# Grading

The following grading structure is used to assess the level of vulnerability found within all Smart Contracts:

| THREAT LEVEL | DEFINITION |
| --- | --- |
| **Critical** | Severe vulnerabilities which compromise the entire protocol and could result in immediate data manipulation or asset loss. |
| **High** | Significant vulnerabilities which compromise the functioning of the smart contracts leading to possible data manipulation or asset loss. |
| **Medium** | Vulnerabilities which if not fixed within in a set timescale could compromise the functioning of the smart contracts leading to possible data manipulation or asset loss. |
| **Low** | Low level vulnerabilities which may or may not have an impact on the optimal performance of the Smart contract. |
| **Informational** | Issues related to coding best practice which do not have any impact on the functionality of the Smart Contracts |

# Description

**LUXPAD TOKEN :** (LXPD) TRUSTLESS PROCESS CREATING A LAUNCHPAD Multi-chain launchpad, create token and NFT in easy way.

**Buy Trading Fees** 4.0% - LP | 1% Task Fee | 1% Marketing | 1% Staking | 2% LP

**Sell Trading Fees** 4.0% - LP | 1% Task Fee | 1% Marketing | 1% Staking | 2% LP

Initial supply : **100,000,000,000 LXPD**

## LUXPAD TOKEN TOKENOMICS

## Luxpad Tokenomics

In this page Luxpad Token information is provided

**Total Supply:** 100,000,000,000 Luxpad Token(LXPD)

**Token Name:** Luxpad Token

**Token Symbol:** LXPD

### For buy/sell

==========

**Marketing Fee:** 1%

**Liquidity Fee:** 2%

**Tax Fee:** 1%

==========

To trade in pancakeswap you must need to change the slippage so can trade/buy/sell

**Slippage Tolerance:** 4%

==========

## Token Allocation:

**Seed Sale:** Round1: 7,000,000,000 LXPD ; unsold token burn and it will send to 0×000000000000000000000000000000000dead.

**Public Launchpad sale:** 15,000,000,000 LXPD no vesting. unsold token burn and it will send to 0×000000000000000000000000000000000dead.

**Liquidity Pool:** 15,000,000,000 LXPD will be use for LXPD LP

**Marketing plan & Exchange listing:** 2,000,000,000 LXPD

**Team & Advisor:** 1,000,000,000 LXPD vesting 16 months, start at 1st month from launch date token is locked for 16 months.

**Community Task Airdrop:** 15,000,000,000 LXPD vesting. 20% TGE, 80% 10 months vesting, Target user 20,000+

**Testnet incentivize reward:** 10,000,000,000 LXPD, 30% TGE, 70% 5 months vesting . how to be qualified go here: **# how-can-be-eligible-for-testnet-event**

**BURN:** WE WILL BURN 35,000,000,000 OF LXPD BEFORE THE PUBLIC SALE

**Vulnerability 0: No important security issue detected.**
**Threat level**: Low

**Description:**
Not a honeypot transaction simulation is success at the moment. Always DYOR before investing.

INFO! There is no liquidity with BNB. Results with non-BNB pair may differ. If the token is not live yet, results may be different once the token is live. It is common for tokens to have 0% taxes before launching on DEX!

**Vulnerability 1: The owner can change the high fee setting function in the contract.**

**Threat level**: Low

**Vulnerability 1:** Gas optimisation

**Threat level 2**: Informational

**Description: this smart-contract can be Modified by Deployer**
This can always change! Do your own due diligence.
INFO! Owner can change trading tax fee. which is Really a normal function for most Smart Contract. Removal fee is private and calculate function

**LUXPAD TOKEN (LXPD)**

No trading data available: either trading is disabled, or no Liquidity for the token Yet.

**Recommendation:**

The contract can be modified so that it can be done via a single call to save gas.

```
891
892     //to recieve ETH from uniswapV2Router when swaping
893     receive() external payable {}
894
895     function _reflectFee(uint256 rFee, uint256 tFee) private {
896         _rTotal = _rTotal.sub(rFee);
897         _tFeeTotal = _tFeeTotal.add(tFee);
898     }
899
900     function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, u
901         (uint256 tTransferAmount, uint256 tFee, uint256 tLiquidity) = _getTValues(tAmount);
902         (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmount, tFe
903         return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tLiquidity);
904     }
905
906     function _getTValues(uint256 tAmount) private view returns (uint256, uint256, uint256)
907         uint256 tFee = calculateTaxFee(tAmount);
908         uint256 tLiquidity = calculateLiquidityFee(tAmount);
909         uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity);
910         return (tTransferAmount, tFee, tLiquidity);
911     }
912
913     function _getRValues(uint256 tAmount, uint256 tFee, uint256 tLiquidity, uint256 current
914         uint256 rAmount = tAmount.mul(currentRate);
915         uint256 rFee = tFee.mul(currentRate);
916         uint256 rLiquidity = tLiquidity.mul(currentRate);
917         uint256 rTransferAmount = rAmount.sub(rFee).sub(rLiquidity);
918         return (rAmount, rTransferAmount, rFee);
919     }
920
921     function _getRate() private view returns(uint256) {
922         (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
923         return rSupply.div(tSupply);
```

## Issues Checking Status

| | Issue description | Checking status |
|---|---|---|
| 1. | Compiler errors. | Passed |
| 2. | Race conditions and Reentrancy. Cross-function race conditions. | Passed |
| 3. | Possible delays in data delivery. | Passed |
| 4. | Oracle calls. | Passed |
| 5. | Front running. | Passed |
| 6. | Timestamp dependence. | Passed |
| 7. | Integer Overflow and Underflow. | Passed |
| 8. | DoS with Revert. | Passed |
| 9. | DoS with block gas limit. | Passed |
| 10. | Methods execution permissions. | Passed |
| 11. | Economy model of the contract. | Passed |
| 12. | The impact of the exchange rate on the logic. | Passed |
| 13. | Private user data leaks. | Passed |
| 14. | Malicious Event log. | Passed |
| 15. | Scoping and Declarations. | Passed |
| 16. | Uninitialized storage pointers. | Passed |
| 17. | Arithmetic accuracy. | Passed |
| 18. | Design Logic. | Passed |
| 19. | Cross-function race conditions. | Passed |
| 20. | Safe Open Zeppelin contracts implementation and usage. | Passed |
| 21. | Fallback function security. | Passed |

## Audit Result

**PASSED**

# Conclusion

During the Vital block Audit process, the LUXPAD TOKEN contract was analysed by manual review and automated testing. All issues identified was after deployment to mainnet. By submitting the contract for audit after Deployment, the team have displayed a strong commitment to security.

Whilst there are no obvious vulnerabilities or security risks identified within the main net contract, it is beyond the scope of this Vital Block Audit to comment upon any risks associated with tokenomics, adoption or platform longevity. Before placing funds in any defi protocol Vital Block encourages potential investors to exercise due diligence and research all projects thoroughly to assess plans for ongoing development and financial sustainability.

# Appendix

## Finding Categories

### Gas Optimization
Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Mathematical Operations
Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Logical Issue
Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Control Flow
Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Volatile Code
Volatile Code findings refer to segments of code that behave unexpectely on certain edge cases that may result in avulnerability.

### Data Flow
Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a structassignment operation affecting an in-memory struct rather than an instorage one.

### Language Specific
Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

### Coding Style
Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

# Appendix

### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

### Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

### Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

### Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

# VitalBlock

Making Defi And Web3 a Safer place

WWW.VITALBLOCK.ORG

Decentralized Smart Contract Auditing Firm.