DEXIRA SMART CONTRACT SECURITY AUDIT

August, 2021

EXECUTIVE SUMMARY

Project Name	DexIRA
Platform	Binance Smart Chain; Solidity
Type	Token Audit
Auditor	Prosper Onah
Language	Solidity
Delivery Date	2021-08-09
Method of audit	OWASP Risk Rating Methodology
Timeline	2 days
Total issues	1
High risk issues	0
Medium risk issues	0
Low risk issues	0
Informational risk issues	1
Undetermined risk issues	0
Specification	https://bscscan.com/address/0x147E07976E1ae78287c33aAFAab87760d32E50A5#code https://github.com/dexIRA/dexIRA/blob/main/dexIRA.sol https://dexira.com/

INTRODUCTION

On 2021-08-06, Prosper Onah performed an audit of the DexIRA smart contracts.

I, Prosper Onah, have no stake in DexIRA. This audit was performed under a paid contract.

This audit was conducted in conformity with OWASP Risk Rating Methodology

DISCLAIMER

Reports do not constitute an “endorsement” or “disapproval” of any project or team, and should not be construed as such. These reports are not intended to be an indicator of the economics or worth of any "product" or "asset" developed by any team or project, and should not be construed as such.

Reports should not be used to make investment or involvement decisions in any way. These reports are not intended to provide investment advice and should not be construed as such.

AUDIT GOALS AND FOCUS

Verification of details

This will verify that every detail in the specification is correctly implemented in the smart contract

Verification of behavior

This will verify that the smart contract does not have any behavior, explicit or implicit that is not captured in the specification.

This audit will also verify that the contract does not violate the original intended behavior of the specification.

Smart Contract Best Practices

This audit will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This audit will evaluate whether the code does what it is intended to do.

Code Quality

This audit will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to either the operators of DexIRA or its users.

Testing and testability

This audit will examine how easily tested the code is, and review how thoroughly tested the code is.

TERMINOLOGY

This audit uses the following terminology.

Likelihood

How likely a bug is to be encountered or exploited

Impact

The impact a bug would have if exploited

Severity

How serious the issue is, derived from Likelihood and Impact as specified by the OWASP risk rating methodology

Severity Level	Explanation
High risk issues	The issue has put the sensitive information of a large number of users at risk, or it is reasonably likely to have a catastrophic impact on the client's reputation or serious financial implications for clients or users.
Medium risk issues	The issue jeopardized a subset of users' sensitive information, would be detrimental to the client's reputation if exploited, or is likely to have a minor financial impact.
Low risk issues	This risk is of low impact.
Informational issues	This does not pose an immediate threat to immediate operations but is relevant for security best practices
Undetermined risk issues	The impact of the issue is uncertain

OVERVIEW

Source code

The smart contract source code was pulled from the smartcontractkit/DexIRA Github repository

Audit Summary

The review was conducted on commit 4f11fadd5f688e38a603107ae973fd1b1311b4c2, which contains the two contracts that are the focus of this review, specifically DexIRA.sol, and DexIRADividendTracker.sol.

Defense against unknown vulnerability, which gives the owner of the contract to respond to future bugs and unknown vulnerabilities includes (not excluding anti-whale):
transferDelayEnabled: line 1283 in the commit above.

PROCEDURE

Manual code review

- Review of the specification and instruction provided to Onah Prosper to make sure I understand the size, scope and functionality of the smart contract.
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerability.
- Comparison to specification to check if the code does what the specifications, source, and instruction provided.

Automated Verification

- To detect certain types of bugs human might not see.
- To free me up to think about bigger picture issues
- Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program.

Toolset

The setup and steps utilized in the process of this audit are outlined below;

- Oyente with Docker.
- Mithril with remixIDE

AUDIT SCOPE

The list of vulnerability checks conducted on the token contract includes but not limited to;

- Reentrancy.
- Access controls
- Arithmetic Issues (integers Overflow/Underflow).
- Unchecked return value for low level call.
- Unsafe external calls
- Centralization of power.
- Business logic contradicting the specification
- Short Address Attack
- Unknown Vulnerability.

SUMMARY OF FINDINGS

ID	Description	Severity
DEX-01	Possible case for **reentrancy** to occur	Information

DETAILED AUDIT REPORT

DEX-01: Possible case for **reentrancy** to occur

Description: Calls to untrusted contracts can introduce several unexpected risks or errors. External calls may execute malicious code in that contract or any other contract that it depends upon. As such, every external call should be treated as a potential security risk. When it is not possible, or undesirable to remove external calls.

Severity: Information.

RECOMMENDATION

Changes after external calls and when interacting with external contracts should be avoided, variables, methods, and contract interfaces should be appropriately stated in a way that makes it clear if interacting with them is potentially unsafe. This applies to private functions that call external contracts.

CONCLUSION

To further conclude all other test made as prescribe in the Scope and goal pass.