BLOCKSTARS TECHNOLOGY

it@blockstars.com.au

SMART CONTRACTS CODE REVIEW AND SECURITY ANALYSIS REPORT

dltx.io

15/10/2021

Table of Contents

INTRODUCTION

This report contains confidential information of audit summary of Forex tokens related smart contract programs running in Giannis_NFT project. It analyses security vulnerabilities, smart contract best practices, and possible attacks, using popular analysis tools and linters. We outlined our systematic approach to evaluate potential security issues in core smart contracts in Giannis_NFT project and provide additional suggestions for improvement.

The basic information of dltx.io Giannis_NFT project

Item	Description
Issuer	dltx.io
Website	https://www.dltx.io/
Source	Smart contract programs (Giannis_NFT)
Language	Solidity
Blockchain	Ethereum
Git Repository	https://github.com/dltxio/Giannis-NFT
Audit method	Static analysis using symbolic execution tools

PROJECT SCOPE AND SPECIFICATIONS

Scope of this project is to identify smart contract vulnerabilities to improve the coding practice in Giannis_NFT (Non-Fungible Token) related smart contracts implemented for dltx.io project. We classified the security vulnerabilities of smart contracts in three categories according to their impact level, such as critical, medium, and low class of vulnerabilities. Moreover, we used the impact level definitions from the recent smart contract security analysis research works [ref1, ref2]. and assigned proper impact values for the identified smart contract vulnerabilities in dltx.io Forex tokens project.

IP1: It raises critical behaviours and attackers can make benefit by using this vulnerability.
IP2: It raises critical behaviours and attackers cannot make benefit using this vulnerability.
IP3: It raises critical behaviours and attackers cannot trigger them externally (If they trigger, they cannot make benefit.
IP4: Contract works normally, and it leads to potential risks of errors when external programs call the contract.
IP5: It works normally, and it will not lead risks for external callers. But there is no re-usability, and it leads to gas wastage.

Impact Levels	Severity
IP1	High Critical
IP2	Moderate Critical
IP3	Average Critical
IP4	Less Critical
IP5	Normal Risk

ANALYSIS RESULT AND SUGGESTIONS

We used different tools and linters to analyse given smart contracts from dltx.io Giannis NFT project. The major tools we used to analyse smart contracts are Mythx (Consensys), and Solhint linter. The platforms we integrated to test the contracts are Remix, Visual Studio Code, Truffle, and Openzepplin Test Environment.

Smart Contract: FixedPriceSale721.sol

FINDING 1

Issue: A floating pragma is set
Severity: **IP5 (Normal Risk)**
Contract: FixedPriceSale721.sol
Description:

- The current pragma Solidity directive is ""^0.8.0"".
- It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.
  Eg: pragma solidity 0.8.0;
- This is especially important if you rely on bytecode-level verification of the code.

Exact place of usage:
pragma solidity ^0.8.0; - Line 2
Smart Contract Weakness Classification: SWC-103

FINDING 2

Issue: Dependence on predictable environment variable
Severity: **IP4 (Less Critical)**
Contract: FixedPriceSale721.sol
Variable name: block.timestamp
Description:

- A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision.
- Avoid using any of those environment variables and be aware that use of these variables introduces a certain level of trust into miners.
- But, It is more critical if you use the block.timestamp values in calculations related to funds.

Exact place of usage:
uint256 currentSaleNumber = (block.timestamp - SALESTART) / saleDuration;
- Lines: 72, 111

Smart Contract Weakness Classification: SWC-116

Smart Contract: ShortTermSale1155.sol

FINDING 1

Issue: State variable visibility is not set.
Severity: IP5 (Normal Risk)
Contract: FixedPriceSale721.sol
Variable Name: idToIpfs
Description:

- It is best practice to set the visibility of state variables explicitly.
- Labelling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.
- The default visibility for "idToIpfs" is internal. Other possible visibility settings are public and private.

Exact place of usage:
mapping(uint256 => string) idToIpfs; - Line 9

Smart Contract Weakness Classification: SWC-108

FINDING 2

Issue: A floating pragma is set
Severity: IP5 (Normal Risk)
Contract: ForexVesting.sol
Description:

- The current pragma Solidity directive is ""^0.8.0"".
- It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.
  Eg: pragma solidity 0.8.0;
- This is especially important if you rely on bytecode-level verification of the code.

Exact place of usage:
pragma solidity ^0.8.0; - Line 2

Smart Contract Weakness Classification: SWC-103

EXECUTIVE SUMMARY

Contracts	Findings & SWC	Status	Action
FixedTermSale721.sol	A floating pragma is set - SWC-103	IP5 (Normal Risk)	Can be ignored
	Dependence on predictable environment variable - SWC-116	IP4 (Less Critical)	It is good to avoid the use of timestamp value. Can be ignored since the timestamp value is not used in fund related calculations.
ShortTermSale1155.sol	State variable visibility is not set - SWC-108	IP5 (Normal Risk)	Can be ignored
	A floating pragma is set - SWC-103	IP5 (Normal Risk)	Can be ignored