SOLIDIFIED

Audit Report for Blue Canyon Investment Ltd. - November 04, 2020

Summary

Audit Report prepared by Solidified covering the ERC721 smart contract and Python scripts provided by Blue Canyon Investment Ltd.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on October 28th, 2020, and the results are presented here.

Responses by the team were received on November 04, 2020. These are included in this report.

Audited Files

The following contracts were covered during the audit:

```
contracts
└── GoldBars.sol
toolbox
├── __init__.py
├── batch.py
├── constants.py
├── deploy.py
├── erc721.py
├── general.py
├── gnosis.py
├── run_example.py
└── testnet_deploy.py
```

Supplied in the following private source code repository:

https://github.com/DuboisGold/dgm-contracts

Notes

The audit covers commit `ca244743411c1a4a2db3164f25602b8f764a379d`

Intended Behavior

The smart contract implements an ERC-721 nonfungible token. A privileged owner can blacklist addresses, freeze transfers, mint, and burn tokens.
The Python scripts provide the functionality to interact with the system and a Gnosis safe acting as a multi-sig contract owner.

SOLIDIFIED

Audit Report for Blue Canyon Investment Ltd. - November 04, 2020

Executive Summary

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	Medium	-
Test Coverage	High	Test coverage not only covers the token itself, but also the more complex interactions with dependencies (Gnosis Safe).

Issues Found

Solidified found that the smart contracts and scripts contain no critical issue, no major issue, 2 minor issues, in addition to 1 informational note.

We recommend all issues are amended, while the notes are up to the team's discretion, as it refers to best practices.

Issue #	Description	Severity	Status
1	Hardcoded gas prices in python scripts	Minor	Acknowledged
2	Compiler Version	Minor	Resolved
3	Notes regarding owner privileges	Note	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. Hardcoded gas prices in python scripts

The gas prices to be used for transactions are hard-coded in the python scripts meant for smart-contract interactions. This can lead to failed or delayed transactions during times of high transaction fees or to excessive cost when gas prices are low.
The fixed price in the current scripts (20 Gwei) is currently too low for mainnet transactions.

Recommendation
Adapt gas prices to be dynamically calculated or to be passed as a parameter.

Team Response
“The functions provided in the toolbox scripts will take the gas price as an argument, for example, execute_gnosis_tx. In actual transactions, we are fetching the gas price from the ethstation APIs and passing it to these functions - both for mainnet and testnet.
We have hardcoded the gas price in the toolbox just for demonstration purposes only.”

2. Compiler Version

The code uses `pragma solidity ^0.6.0;` to define the compiler version to be used. Whilst we do not recommend using the latest compiler version due to the risk of undiscovered bugs, versions used in the Solidity 0.6 range should be 0.6.10 or newer since a number of important bug fixes have been applied.

Recommendation
Lock the pragma to use the latest 0.6 Solidity release: `pragma solidity 0.6.12;`

Update: The team has implemented the recommendation

Notes

3. Notes regarding owner privileges

The smart contract system relies on a privileged admin role (owner of the token smart contract). The owner can mint and burn tokens, freeze transactions and blacklist addresses.
Whilst this is in line with the business requirements, this note is intended to make users aware of the custodian nature of the system.
Ownership is intended to be assigned to a Gnosis Safe smart contract for a multi-signature procedure for privileged admin processes. Users are advised to confirm this on the actual deployment.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of provided by Blue Canyon Investment Ltd. or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.