SOLIDIFIED

Audit Report for Immutable - April 13, 2021

Summary

Audit Report prepared by Solidified covering the Guild Of Guardian token and presale smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on 31 March 2021, and the results are presented here.

Audited Files

The source code has been supplied in the form of an archive file:

hardhat-final.tar.gz

SHA256: `a7cb88a5a1d4ad261a7f313da9aad47083dbe5a74ea3febe478470b229af5ab5`

UPDATE: Fixes have been provided in the file:

hardhat-final-post-audit-changes.tar.gz

SHA256: `2991d4b8c4022464f555580c1c38341bc26ae92afd97be7ce716782cf76395f7`

The scope of the audit was limited to the following files:

```
contracts
├── Constants.sol
├── Dice.sol
├── ExchangeRate.sol
├── GuardiansToken.sol
├── GuildOfGuardiansPreSale.sol
├── GuildOfGuardiansPreSaleTestable.sol
├── Inventory.sol
├── Referral.sol
├── Treasury.sol
├── UniswapV2PairTestable.sol
└── interfaces
    └── IUniswapV2Pair.sol
```

Intended Behavior

The smart contracts implement a presale for the Guild Of Guardian project.

Note, that code only implements the purchase functionality and emits events. The actual asset issuance is performed by Immutable in a trusted setup separately.

Code Complexity and Test Coverage

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 equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium-Low	-
Code readability and clarity	Medium-High	-
Level of Documentation	Medium	-
Test Coverage	High	-

Test coverage report:

```
---------------------------------------|----------|----------|----------|----------|----------------|
File                                   |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
---------------------------------------|----------|----------|----------|----------|----------------|
 contracts/                            |    85.62 |    92.11 |    62.96 |    85.94 |                |
  Constants.sol                        |      100 |      100 |      100 |      100 |                |
  Dice.sol                             |    84.62 |    83.33 |       75 |    84.62 |          63,75 |
  ExchangeRate.sol                     |      100 |      100 |      100 |      100 |                |
  GuardiansToken.sol                   |        0 |      100 |        0 |        0 |              8 |
  GuildOfGuardiansPreSale.sol          |      100 |      100 |      100 |      100 |                |
  GuildOfGuardiansPreSaleTestable.sol  |       25 |      100 |     87.5 |       28 |... 58,59,60,61 |
  Inventory.sol                        |    98.84 |    92.19 |      100 |    98.88 |        585,591 |
  Referral.sol                         |      100 |      100 |      100 |      100 |                |
  Treasury.sol                         |      100 |      100 |      100 |      100 |                |
  UniswapV2PairTestable.sol            |     12.5 |      100 |     7.14 |     12.5 |... 127,131,139 |
 contracts/interfaces/                 |      100 |      100 |      100 |      100 |                |
  IUniswapV2Pair.sol                   |      100 |      100 |      100 |      100 |                |
---------------------------------------|----------|----------|----------|----------|----------------|
All files                              |    85.62 |    92.11 |    62.96 |    85.94 |                |
---------------------------------------|----------|----------|----------|----------|----------------|
```

Issues Found

Solidified found that the Guild of Guardian sales contracts contain no critical issues, 2 major issues, 2 minor issues, in addition to 5 informational notes.

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

Issue #	Description	Severity	Status
1	Dice.sol: Insecure Random Number Generation	Major	Acknowledged
2	User can use any address as referrer to get discount	Major	Acknowledged
3	Treasury.sol, Inventory.sol and Referral.sol: Use call() instead of transfer()	Minor	Resolved
4	Dice.sol: Function _getSecondDiceRoll() is not bound by maxDiceRoll	Minor	Resolved
5	GuardiansToken.sol: Unimplemented initial token distribution	Note	-
6	Use constants instead of magic numbers	Note	-
7	Inventory.sol: Length of arrays as function parameters is not enforced	Note	-

Critical Issues

No critical issues have been found.

Major Issues

1. Dice.sol: Insecure Random Number Generation

The `Dice` contract implements an optimized commit / reveal RNG scheme. However, implementation fails to enforce a commitment to the future block number in the commit phase. The caller can arbitrarily choose and pre-calculate a suitable `_commitBlock` argument when calling `getSecondDiceRoll()` since there is nothing in the code that enforces `_commitBlock` to equal the block number of the call to `getFirstDiceRoll()` (or any other specific block number).

Recommendation
Enforce a specific commit block in the commit phase.

Team Response
"Immutable has made a business decision to build this in a way that is publically transparent, verifiable, auditable, but not fully decentralised. The company can be a trusted party, as long as it would be possible for a third party to verify its actions. This is done for a few reasons:

- *High gas costs of fully decentralised solutions likely to be a barrier in the context of playing games / buying in game items*
- *Immutable has built up significant trust with existing products, and would have a lot to lose by acting inappropriately*

So in this case, the company will be trusted to specify the correct block when calling getSecondDiceRoll. While the company could enter an older block, anyone can verify this was incorrect by looking at which block that particular order / first dice roll occurred in."

2. Inventory.sol: User can use any address as referrer to get discount

Users can use any (or their own address) as the referrer while calling the method `purchase` to get a referrer bonus and discount.

Recommendation
It is recommended to track the referrer address or use some whitelisting method to efficiently distribute referrals or give discounts.

Team Response
“This is intentional to minimise gas usage. Immutable is fine with people entering any address in order to get a 5% discount.”

Minor Issues

3. Treasury.sol, Inventory.sol and Referral.sol: Use call() instead of transfer()

The functions `withdraw()`, `purchase()` and `withdrawBonus()` use the `transfer()` function to transfer ETH to `msg.sender`. However, gas prices were changed in the Istanbul hard fork, meaning that the gas stipend forwarded to `msg.sender` may not be enough for smart contract receivers to do basic bookkeeping.

For a more in depth discussion of issues with `transfer()` and smart contracts, please refer to:
https://diligence.consensys.net/blog/2019/09/stop-using-soliditys-transfer-now/

Recommendation

Use the following pattern for transferring ETH:

```
(bool success, ) = recipient.call{value:amount}("");
require(success, "Transfer failed.");
```

4. Dice.sol: Function _getSecondDiceRoll() is not bound by maxDiceRoll

Values returned by `_getSecondDiceRoll()` can be greater than `maxDiceRoll`.

Recommendation

Mod the function’s calculation by `maxDiceRoll` in order to stay within the maximum range.

Informative Notes

5. `GuardiansToken.sol`: Unimplemented initial token distribution

The contract’s constructor mints all tokens to the deployer and is marked with the following comment:

`// TODO Set initial token distribution`

Recommendation

Complete the implementation.

6. Use constants instead of magic numbers

The code uses hardcoded values in several places. For example, `for (uint256 i = 0; i < 9; i++)` is using the hardcoded length instead of using the constants defined already. It is commonly considered best practice to use constant instead of hardcoded numbers.

Recommendation

Remove the magic number and replace it with constants.

7. `Inventory.sol`: Length of arrays as function parameters is not enforced

The methods `_enforceOrderLimits` and `addStock` are using a constant to loop over the array parameter and access its elements. Since the array parameter length can be anything, this approach will throw unexpected errors if the array length is smaller than the constant used to iterate.

Recommendation

Add `require` statements enforcing the correct length.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Immutable 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.