DoubleDice Audit

March 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the DoubleDice project. First we will provide a summary of our discoveries and then we will show the details of our findings.

Scope

The contracts audited are from the `https://github.com/DoubleDice-com/doubledice-platform` git repository. The first iteration of this audit is based on the commit `5bd440da26ebb258698d684ac2281f38aa607c32`.

Fixes were checked on commits `de79f3d4741f34a0a1b45ae049e27a33d9a88516`, `6def4aa8a3a8d582bd9943e772bfe3f0a2c2671f`, `b5a38f2d8d6f9958af906c69d627a8b5f9ab4439`, `e0fca9c9f1eaa090b0c9398ea0fdc9f7da9114ea`, `09c5c068042125b8c42f8be9f1d9a2b6b5efada6`, and `41327e967c5a644e405480b4c3750f8919bb0026`.

The second iteration of this audit is based on the commit `3fea8d3a280a06b14c777b180ca9a3b8f4950587`. Fixes were checked in commit `b2c94d7798668cb368993dac3fd0ca74a21002fe`.

The audited contracts are:

- `contracts/BaseDoubleDice.sol`: core VF lifecycle
- `contracts/ForkedERC1155UpgradeableV4_5_2.sol`: Multi-token implementation fork
- `contracts/ChallengeableCreatorOracle.sol`: Challengeable VF resolver
- `contracts/SimpleOracle.sol`: Simple VF resolver
- `contracts/CreationQuotas.sol`: VF Quota limiter
- `contracts/DoubleDice.sol`: Top contract
- `contracts/VirtualFloorMetadataValidator.sol`: VF parameters validator
- `contracts/MultipleInheritanceOptimization.sol`: Optimization
- `library/FixedPointTypes.sol`: Fixed types implementation
- `library/VirtualFloors.sol`: VF main contract

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found 1 critical issue, 2 medium issues and a minor issue which were fixed by the development team. Also, several enhancements were proposed.

Security Issues

ID	Title	Severity	Status
CR-01	Block.timestamp Manipulation	Critical	Resolved
ME-01	Insecure Token Transfer	Medium	Resolved
ME-02	Reentrancy Point in `BaseDoubleDice.sol`	Medium	Resolved
MI-01	Floating Pragma	Minor	Resolved

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

OPERATOR_ROLE

This role cuts across various contracts (`BaseDoubleDice`, `ChallengeableCreatorOracle` and `CreationQuotas`)

An address with the operator role can:
- Change the internal state of a VF from `Active` to `Claimable_Refunds_Flagged`.
- End challenge when it is spawned with a final outcome.
- Adjust Creation Quotas.
- Set the result if the VF creator does not do so.

BaseDoubleDice

DEFAULT_ADMIN_ROLE

This role is allowed to pause/unpause the contract, configure platform parameters.

ChallengeableCreatorOracle

VF_CREATOR

This role is not defined as such, but the only address that can execute the function `setResult()`, line 132, is the creator of the VF.

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

CR-01 Block.timestamp Manipulation

Location:

- `contracts/BaseDoubleDice.sol:228,318`
- `contracts/ChallengeableCreatorOracle.sol:107-156`
- `library/VirtualFloors.sol:23,31,83`

Malicious miners can manipulate the block's timestamp value to gain advantages, so a developer can't rely on the preciseness of the provided timestamp. This is particularly important in short-duration betting events, where the variation of this value may be enough to manipulate them. For more information see SWC-116 (https://swcregistry.io/docs/SWC-116).

Recommendation

A way to mitigate the lack of precision of this value in betting events is to make sure the event is closed some minutes before the actual event, so the bets cannot be manipulated.

Status

Fixed on commits: `b5a38f2d8d6f9958af906c69d627a8b5f9ab4439`, `09c5c068042125b8c42f8be9f1d9a2b6b5efada6`, and `41327e967c5a644e405480b4c3750f8919bb0026`.

Medium Severity Issues

ME-01 Insecure Token Transfer

Location:

- `contracts/BaseDoubleDice.sol:527,562`

The ERC20 `transfer()` function does not always revert in the case of error, sometimes it just returns false. In this particular contract the code does not check for the return value, thus ignoring if the transfer was successful or not.

Recommendation

Replace the offending function with the safer `safeTransfer()` function that automatically asserts in the case of error.

Status

Fixed on commit: `e0fca9c9f1eaa090b0c9398ea0fdc9f7da9114ea`

ME-02 Reentrancy Point in `BaseDoubleDice`.sol

Location:

- `contracts/BaseDoubleDice.sol:310`

The function `commitToVirtualFloor()` in the `BaseDoubleDice.sol` contract is susceptible to reentrancy when calling `token.safeTransferFrom()`. Notice that if the attacker controls the token contract, he can implement reentrancy attacks in the `safeTransferFrom()` call at `BaseDoubleDice.sol:310` function by calling `commitToVirtualFloor()` recursively. The attacker can use the reentrancy to skip the update of the outcomeTotals variable at `baseDoubleDice.sol:331.`

Recommendation

Use a reentrancy guard or place reentrancy checks.

Status

Reentrancy guards added, additionally all other reentrancy risks were addressed on commit `6def4aa8a3a8d582bd9943e772bfe3f0a2c2671f`.

Minor Severity Issues

MI-01 Floating Pragma

Location:

- `contracts/ForkedERC1155UpgradeableV4_5_2.sol:4`

Contracts should be deployed with the same compiler version that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

Lock the pragma version, replacing `pragma solidity ^0.8.0;` with a specific patch, preferring the most updated version. For example, `pragma solidity 0.8.12;`.

Status

Fixed on commit `de79f3d4741f34a0a1b45ae049e27a33d9a88516`.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Table

ID	Title	Status
EN-01	Revert transactions as soon as possible	Implemented
EN-02	Not Default Action With Invalid States	Implemented

Details

EN-01 Revert Transactions As Soon As Possible

Location:

- `doubledice-platform/contracts/BaseDoubleDice.sol:492`

In the `_resolve()` function, some actions are first performed and then it is queried if the contract is paused.

Recommendation

It would be better to first check if the contract is paused, so that the gas expense is lower.

Status

Implemented. (commit: `fb09655d9b0a994cf12ec71476cb01117a0ecc55`).

EN-02 Not Default Action With Invalid States

Location:

- `doubledice-platform/contracts/library/VirtualFloors.sol:49`

The final else of the `state()` function returns the value `Claimable_Refunds_Flagged`. Currently it is not a bug, since it uses all the values of the `VirtualFloorInternalState` enum, but if an extra value is added in the future, it could lead the contract to return incorrect states.

Recommendation
Use else if with the condition and then put a revert or assert on the else.

Status
Implemented. In the else statement it is validated with an assert.
(commit: `fb09655d9b0a994cf12ec71476cb01117a0ecc55`).

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest for other stakeholders of the project, including users of the audited contracts, owners or project investors.

Centralization

In the Privileged Roles section, the `DEFAULT_ADMIN_ROLE, OPERATOR_ROLE` and `VF_CREATOR` system actors were described.
`DEFAULT_ADMIN_ROLE` has the capability to configure/change various parameters of the platform such as fees and beneficiary address. It can also pause/unpause the contract. Particularly within these functionalities that can be paused, is the claim for payments and refunds.
Each `VF_CREATOR` centralizes setting its own result.
Finally, `OPERATOR_ROLE` can change the internal state of a VF from Active to Claimable_Refunds_Flagged, end challenge when it is spawned with a final outcome and set the result if the VF creator does not do so.

Changelog

- 2022-03-15 – Initial report based on commit `5bd440da26ebb258698d684ac2281f38aa607c32`.
- 2022-03-22 – Fixes checked on commits `de79f3d4741f34a0a1b45ae049e27a33d9a88516`, `6def4aa8a3a8d582bd9943e772bfe3f0a2c2671f`, `b5a38f2d8d6f9958af906c69d627a8b5f9ab4439`, `e0fca9c9f1eaa090b0c9398ea0fdc9f7da9114ea`, `09c5c068042125b8c42f8be9f1d9a2b6b5efada6`, and `41327e967c5a644e405480b4c3750f8919bb0026`.
- 2022-03-28 – Second iteration of this audit, based on commit `3fea8d3a280a06b14c777b180ca9a3b8f4950587`.
- 2022-03-31 – Fixes checked on commit `b2c94d7798668cb368993dac3fd0ca74a21002fe`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the DoubleDice project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.