Owldinal
Audit Report

Tue Apr 23 2024

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

Owldinal Audit Report

1 Executive Summary

1.1 Project Information

Description	The 1st native NFT gaming protocol on MerlinLayer2 begins with Gen0, featuring 999 rare and magical pet Owls.
Type	Game
Auditors	ScaleBit
Timeline	Thu Mar 28 2024 - Tue Apr 23 2024
Languages	Solidity
Platform	Merlin Chain
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Owldinal/OwlGame
Commits	3d2b9b6a3e6598ee38a5d84692bb3ccf1251a8c7 348e75fa21fa349795000d1a6aa35f8dc11fc64a

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
OGA	contract/contracts/OwlGame.sol	90392df1cec9a8a43a1c1753eb7a7cb7536c0875
OGA	contract/contracts/OwlGame.sol	ce59063116d85c965df769327f5483e0afd7e207

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	8	7	1
Informational	1	1	0
Minor	3	3	0
Medium	2	1	1
Major	2	2	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Owldinal to identify any potential issues and vulnerabilities in the source code of the Owldinal smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 8 issues of varying severity, listed below.

ID	Title	Severity	Status
OGA-1	Variables are Used Without Initialization	Major	Fixed
OGA-2	Staking Without Locking the User's NFT	Major	Fixed
OGA-3	Incompatible With Deflationary Token	Medium	Acknowledged
OGA-4	Pseudo-random in `_generateInviteCode()`	Medium	Fixed
OGA-5	Unused Constant	Minor	Fixed
OGA-6	Division Before Multiplication Will Cause Precision Loss	Minor	Fixed
OGA-7	Cache Array Length Outside of Loop	Minor	Fixed
OWL-1	May Add the `nonReentrant` Modifier	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Owldinal Smart Contract :

Admin

- The admin can add pool price by the `addPrize` function.
- The admin can enable or disable the boost by the `setMoonBoost` function.
- The admin can withdraw all of the tokens by the `withdraw` function.
- The admin can update the owl reward every 4 hours by the `updateAllFruitRewards` function.
- The admin can mint a MysteryBox token by the `mintMysteryBox` function.

User

- The user can get the mint request by the `requestMint` function.
- The user can stake the owldinal NFT by the `stakeOwldinalNft` function.
- The user can stake the MysteryBox token by the `stakeMysteryBox` function.
- The user can unstake the owldinal NFT by the `unstakeOwldinalNft` function.
- The user can unstake the MysteryBox token and claim the rewards by the `claimAndUnstakeMysteryBox` function.
- The user can claim the invite rewards by the `claimInviterReward` function.

4 Findings

OGA-1 Variables are Used Without Initialization

Severity: Major

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#53,54,55

Descriptions:

The `OwlGame` contract declares instances of `OwlToken`, `MysteryBoxGen0`, and `MysteryBoxGen1` contracts, but none of these instances are initialized, which means that the `owlToken`, `boxGen0Contract`, and `boxGen1Contract` variables have the value of zero, and all subsequent calls are also calls to zero addresses.

Suggestion:

It is recommended to initialize these parameters before using them.

Resolution:

The client has resolved the issue.

OGA-2 Staking Without Locking the User's NFT

Severity: Major

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#148-211

Descriptions:

In the `stakeBox` function, there is no requirement to pass the user's NFT to the `OwlGame` contract address, whether it is to pledge `boxGen0` tokens or `boxGen1` tokens, which results in the user being able to call the `stakeBox` function arbitrarily, e.g., if the `boxGen1` tokens are pledged, multiple calls will result in the `fruitIdList` or `elfIdList` containing multiple identical tokenId, which leads to the user being able to receive multiple rewards with a single tokenId and a larger percentage of rewards can be released by increasing the number of `fruitIdLists`. In addition to that, in the case of pledging `boxGen0` tokens, the caller is asked to transfer the tokenId number of `owlTokens` to the `OwlGame` contract address, would like to confirm that this is not an error.

Suggestion:

It is recommended to confirm that this is in line with the design philosophy.

Resolution:

The client has resolved the issue.

OGA-3 Incompatible With Deflationary Token

Severity: Medium

Status: Acknowledged

Code Location:

contract/contracts/OwlGame.sol#90-96

Descriptions:

In the `addPrize` function, due to the unknown address of `owlToken`, when the token is deflationary, the number of tokens transferred to the contract by the user may not be accurate.

Suggestion:

Since it's not known exactly what type of token this is, it's recommended to confirm whether such a question would conflict with the design philosophy.

Resolution:

The contract will not use mismatched tokens such as deflation tokens.

OGA-4 Pseudo-random in `_generateInviteCode()`

Severity: Medium

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#380-397

Descriptions:

In the `_generateInviteCode()` method, the value returned by the `generateRandomNumber()` method is not a truly random number, instead, it is a deterministic value calculated based on predictable values such as `block.number`, `msg.sender`, `block.coinbase`, `block.prevrandao` and so on. So when a user joins the game for the first time, they can set themselves up as an invitee, it doesn't seem right.

Suggestion:

It's recommended to change a different random number implementation, such as using Chainlink, etc.

Resolution:

The client has resolved the issue.

OGA-5 Unused Constant

Severity: Minor

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#30,44

Descriptions:

There are unused variables in the contract.

```
uint256 totalRebateEarned;
```

```
struct Invite {
    address inviter;
    address invitee;
    bool isSet;
}
```

Suggestion:

It is recommended to remove unused variables if there's no further design.

Resolution:

The client has resolved the issue.

OGA-6 Division Before Multiplication Will Cause Precision Loss

Severity: Minor

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#307

Descriptions:

In the `updateAllFruitRewards` function, division before multiplication is used to calculate how many rewards to release. However, this is a bad practice and will result in calculating `totalRewards` precision loss.

Suggestion:

To minimize precision loss, it is advisable to consider changing the calculation sequence by performing multiplication before division. This ensures that the precision of the result is as high as possible before the division operation, thereby avoiding precision losses.

Resolution:

The client has resolved the issue.

OGA-7 Cache Array Length Outside of Loop

Severity: Minor

Status: Fixed

Code Location:

contract/contracts/OwlGame.sol#238-244

Descriptions:

If not cached, the solidity compiler will always read the length of the array during each iteration. That is, if it is a storage array, this is an extra sload operation (100 additional extra gas for each iteration except for the first) and if it is a memory array, this is an extra mload operation (3 additional gas for each iteration except for the first).

Suggestion:

It is recommended to cache the array length before entering the loop.

Resolution:

The client has resolved the issue.

OWL-1 May Add the `nonReentrant` Modifier

Severity: Informational

Status: Fixed

Code Location:

contract/contracts/Owldinal.sol#313,350,396,420,562

Descriptions:

The contract adds the `nonReentrant` modifier to the `requestMint` function to prevent reentry but does not add it in other functions.

Suggestion:

It is recommended to add the `nonReentrant` modifier to all publicly called functions to avoid risks.

Resolution:

The client has resolved the issue.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.