# BlockLords
# Audit Report

Mon Apr 29 2024

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

ScaleBit

# BlockLords Audit Report

## 1 Executive Summary

### 1.1 Project Information

| | |
|---|---|
| Description | Blockchain Games |
| Type | Game |
| Auditors | ScaleBit |
| Timeline | Tue Apr 16 2024 - Tue Apr 23 2024 |
| Languages | Solidity |
| Platform | Ethereum |
| Methods | Architecture Review, Unit Testing, Manual Review |
| Source Code | https://github.com/blocklords/dynasty-smart-contract |
| Commits | ffa192d1ec152b65cbc359baef2adc693d8f290e<br>9408abc9c982bdcd0d028aedf2ec1442950c052c<br>c9c24adcab1274ec20ab1ade00cdfe788f52d5cc<br>07ec7744cfc137a5c0190e3301fc579a7b9895b9 |

## 1.2 Files in Scope

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

| ID | File | SHA-1 Hash |
|---|---|---|
| HNFT | contracts/nfts/HouseNFT.sol | 94bdb6f004f00b07748c502fc1e3d14cbc0a916b |
| BNFT | contracts/nfts/BannerNFT.sol | 3e21bc029a3f2c7cc306df54a40a25e693a3293e |
| HNFT1 | contracts/nfts/HeroNFT.sol | ef0126d003ceccef9807338f0b6e7e43b74bb4cd |
| MAR | contracts/marketplace/Marketplace.sol | 47085918b5f1ef96aefdebf9cdc17ba7d09c20db |
| MIS | contracts/game/Missions.sol | 771794803a3c7a3489173cb3935c405e84548075 |
| DUE | contracts/game/Duel.sol | abe1c4c3f110abbd1395b30e1142824ec32ea92d |

## 1.3 Issue Statistic

| Item | Count | Fixed | Acknowledged |
|---|---|---|---|
| Total | 7 | 7 | 0 |
| Informational | 3 | 3 | 0 |
| Minor | 3 | 3 | 0 |
| Medium | 0 | 0 | 0 |
| Major | 1 | 1 | 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 BlockLords to identify any potential issues and vulnerabilities in the source code of the BlockLords 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 7 issues of varying severity, listed below.

| ID | Title | Severity | Status |
|---|---|---|---|
| HNF-1 | Incomplete Signature Verification | Minor | Fixed |
| MAR-1 | Improperly Designed Buy Function | Major | Fixed |
| MAR-2 | Business logic | Minor | Fixed |
| MAR-3 | Constructor Missing Check | Informational | Fixed |
| MAR-4 | Lack of Events Emit | Informational | Fixed |
| MIS-1 | Nft Lock | Minor | Fixed |
| MIS-2 | Redundant Code | Informational | Fixed |

# 3 Participant Process

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

**contracts/nfts/BannerNFT.sol**
**Deployer**

- Deployer determines the contract `Owner` address and `verifier` address when deploying the contract.

**Owner**

- Owner can use the `setVerifier` function to set the `verifier` address for signature verification.
- Owner can set the `baseUri` address through the `setBaseURI` function.

**USER**

- User can mint BannerNFT’s NFT through the `safeMint` function, which requires a `verifier` address signature.
- User can burn BannerNFT’s NFT through the `Burn` function.

**contracts/nfts/HeroNFT.sol**
**Deployer**

- Deployer determines the contract `Owner` address and `verifier` address when deploying the contract.

**Owner**

- Owner can use the `setVerifier` function to set the `verifier` address for signature verification.
- Owner can set the `baseUri` address through the `setBaseURI` function.

**USER**

- User can mint HeroNFT’s NFT through the `safeMint` function, which requires a `verifier` address signature.
- User can burn BannerNFT’s NFT through the `Burn` function.

**contracts/nfts/HouseNFT.sol**
**Deployer**

- Deployer determines the contract `Owner` address, `verifier` address, and `heroNft` address when deploying.

**Owner**

- Owner can use the `setVerifier` function to set the `verifier` address for signature verification.
- Owner can set the `baseUri` address through the `setBaseURI` function.
- Owner can set the `heroNft` address through the `setHeroNft` function.

**USER**

- User can mint HouseNFT’s NFT through the `safeMint` function, which requires a `verifier` address signature.
- User can set the parameters of `house` through the `setHouse` function, `flagShape` , `houseSymbol` , `flagColor` , `houseName` , `lordNftId` , which requires `verifier` address signature and owner verification of houseId NFT.

**contracts/game/Duel.sol**

**Deployer**

- Deployer determines the contract `Owner` address, `verifier` address and `heroNft` address when deploying the contract.

**Owner**

- Owner can use the `setVerifier` function to set the `verifier` address for signature verification.
- Owner can pause/unpause the contract through the `pause/unpause` function.

**Player**

- Player can start the Duel game through the `startDuel` function, which requires a `verifier` address signature and Player’s NFT (heroNft).
- Player can complete the Duel game through the `finishDuel` function, which requires a `verifier` address to sign Player’s NFT (heroNft).

**contracts/game/Missions.sol**

**Deployer**

- Deployer determines the contract `Owner` address, `verifier` address and `heroNft` address when deploying.

**Owner**

- Owner can use the `setVerifier` function to set the `verifier` address for signature verification.
- Owner can pause/unpause the contract through the `pause/unpause` function.

**Player**

- Player can start the Duel game through the `startDuel` function, which requires a `verifier` address signature and Player's NFT (heroNft).
- Player can complete the Duel game through the `finishDuel` function, which requires a `verifier` address to sign Player's NFT (heroNft).

**contracts/marketplace/Marketplace.sol**

**Deployer**

- Deployer determines the contract `Owner` address, `feeReceiver` address and `feeRate` when deploying.

**Owner**

- Owner can set `salesEnabled` through `enableSales` function.
- Owner can set the `supportedNft` address through the `addSupportedNft/removeSupportedNft` function.
- Owner can pass the `addSupportedCurrency/removeSupportedCurrency` function `supportedCurrency` address.
- Owner can set the `feeReceiver` address through the `setFeeReceiver` function.
- Owner can set `feeRate` through `setFeeRate` function, but it will not exceed 10%.

**USER**

- User can create an order to sell NFT through the `sell` function, specifying the price and token type, the `_nftAddress` address and the `_currency` support address set by the Owner.
- User can close the NFT order to be sold through the `cancelSell` function.
- User can purchase NFT through the `buy` function, and the settlement fee will be sent to `feeReceiver`.

# 4 Findings

## HNF-1 Incomplete Signature Verification

Severity: Minor

Status: Fixed

Code Location:

contracts/nfts/HeroNFT.sol;

contracts/nfts/BannerNFT.sol;

contracts/nfts/HouseNFT.sol;

contracts/game/Missions.sol;

contracts/game/Duel.sol

Descriptions:

The signature verification function lacks verification of expiration time, and the verification is not rigorous enough.

```
bytes32 message        = keccak256(abi.encodePacked(_to, _tokenId, address(this),
nonce[_to]));
```

Suggestion:

It is recommended to add a `deadline` and `chainID`.

Resolution:

Fixed as suggested, added `deadline` and `chainID`.

```
bytes32 message        = keccak256(abi.encodePacked(_nftId, _from, address(this),
nonce[_from], _deadline, block.chainid));
```

## MAR-1 Improperly Designed Buy Function

Severity: Major

Status: Fixed

Code Location:

contracts/marketplace/Marketplace.sol#189

Descriptions:

When a user pays ETH and currency token at the same time, the funds will be locked in the contract.

```
    if (obj.currency == address(0x0)) {
        require (msg.value >= price, "your price is too low");
        uint256 returnBack = msg.value - price;
        if (returnBack > 0)
            payable(msg.sender).transfer(returnBack);
        if (tipsFee > 0)
            feeReceiver.transfer(tipsFee);
        obj.seller.transfer(purchase);
    } else {
        IERC20(obj.currency).safeTransferFrom(msg.sender, feeReceiver, tipsFee);
        IERC20(obj.currency).safeTransferFrom(msg.sender, obj.seller, purchase);
    }
```

Suggestion:

It is recommended to add judgment to handle each situation individually.

Resolution:

Check added.

```
else {require(msg.value == 0, "invalid value");...}
```

## MAR-2 Business logic

Severity: Minor

Status: Fixed

Code Location:

contracts/marketplace/Marketplace.sol#217

Descriptions:

There are no restrictions on seller/buyer in the contract, which may lead to some abnormal call chains. For example, through the `safeTransferFrom` in the `buy` function, the `onERC721Received` function is used to control the status when the status has not changed. For example, the attacker can create it through re-entry. 3 contracts are cross traded.

Suggestion:

It is recommended to add the judgment `require(tx.origin == msg.sender);` to ensure that only game player accounts are called  or add modifier `nonReentrant`.

Resolution:

Fix as suggested add modifier `nonReentrant`, and the code execution order was modified according to Checks-Effects-Interactions..

```
function startDuel(address _from, uint256 _nftId, uint256 _deadline, uint8 _v, bytes32 _r,
bytes32 _s) external nonReentrant
```

## MAR-3 Constructor Missing Check

Severity: Informational

Status: Fixed

Code Location:

contracts/marketplace/Marketplace.sol#56,57

Descriptions:

The feeReceiver address in the `constructor` function lacks a 0 address check, and feeRate can be set to more than 10% in the constructor.

```
  constructor(address initialOwner, address payable _feeReceiver, uint256 _feeRate)
Ownable(initialOwner) {
      feeReceiver = _feeReceiver;
      feeRate = _feeRate;
      // initReentrancyStatus();
  }
```

Suggestion:

It is recommended to increase the require.

Resolution:

Fix as suggested.

```
      require(_feeReceiver != address(0), "receiver address should not be equal to 0");
      require(_feeRate <= 100, "fee rate can not exceed 10%");
```

## MAR-4 Lack of Events Emit

Severity: Informational

Status: Fixed

Code Location:

contracts/marketplace/Marketplace.sol#67,73,80,87,94,101,108

Descriptions:

The module lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues. For example, `enableSales` , `addSupportedNft()` , `addSupportedCurrency` , and so on.

```
  function setHeroNFT(address _nftAddress) external onlyOwner{
      require(_nftAddress != address(0), "nft address can't be zero address ");
      heroNft = _nftAddress;
  }
```

Suggestion:

It is recommended to emit events for those important functions.

Resolution:

Fix as suggested.

```
  event EnableSales(bool indexed enableSales, uint256 indexed time);
  event AddSupportedNft(address indexed nftAddress, uint256 indexed time);
  event RemoveSupportedNft(address indexed nftAddress, uint256 indexed time);
  event AddSupportedCurrency(address indexed currencyAddress, uint256 indexed
time);
  event RemoveSupportedCurrency(address indexed currencyAddress, uint256 indexed
time);
  event SetFeeReceiver(address indexed feeReceiver, uint256 indexed time);
  event SetFeeRate(uint256 indexed rate, uint256 indexed time);
```

## MIS-1 Nft Lock

Severity: Minor

Status: Fixed

Code Location:

contracts/game/Missions.sol#91;

contracts/game/Duel.sol#71

Descriptions:

Owner can set the `heroNFT` address through `setHeroNFT` function. When there are `duel/missions` in progress, the game player's NFT cannot be transferred out.

Suggestion:

It is recommended that `heroNFT` is only set up once.

Resolution:

The `setHeroNFT` function has been deleted, and the `heroNFT` address cannot be changed after initialization is confirmed.

## MIS-2 Redundant Code

Severity: Informational

Status: Fixed

Code Location:

contracts/game/Missions.sol#44;

contracts/marketplace/Marketplace.sol#217

Descriptions:

Verify logical conflict, `playerTeams[_from][teamId][i] == 0` is the priority condition in the loop, `if (playerTeams[_from][teamId][i] != 0)` will not be used in `startMissions` function is executed.

```
if (playerTeams[_from][teamId][i] != 0) {
require(IERC721(heroNft).ownerOf(nftIds[i]) == _from, "hero NFT does not belong to
sender");
}
```

The buyer is declared as payable in `struct SalesObject`, but there is no code to use the payable attribute, only the value of Buyer is assigned.

```
obj.buyer = payable(msg.sender);
```

Suggestion:

It is recommended to remove the redundant code.

Resolution:

Fix as suggested.

# 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