our shielding · Your smart contracts, our shielding · Your smart c

shieldify

# GojuoNFT and ZKojuoNFT

SECURITY REVIEW

Date: 13 August, 2023

# CONTENTS

## 1. About Shieldify

We are Shieldify Security – a company on a mission to make web3 protocols more secure, cost-efficient and user-friendly. Our team boasts extensive experience in the web3 space as both smart contract auditors and developers that have worked on top 100 blockchain projects with multi-million dollars in market capitalization.

Book an audit and learn more about us at shieldify.org or @ShieldifySec

## 2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

## 3. About GojuoNFT and ZKojuoNFT

GojuoNFT and ZKojuoNFT are collections of 8112 uniquely generated poker cards with Japanese characters in an ERC-721 standard issued on both Ethereum, Polygon & zkSync Era. Unlike most NFT projects that only store a website link into the blockchain, all their visual elements and all metadata are securely put on the Ethereum, Polygon & zkSync Era blockchain, making these collections truly immutable.

Learn more about GojuoNFT's concept and the technicalities behind it here.

## 4. Risk Classification

| Severity | Impact: High | Impact: Medium | Impact: Low |
|---|---|---|---|
| **Likelihood: High** | Critical | High | Medium |
| **Likelihood: Medium** | High | Medium | Low |
| **Likelihood: Low** | Medium | Low | Low |

### 4.1 Impact

- **High** – results in a significant risk for the protocol's overall well-being. Affects all or most users
- **Medium** – results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** – losses will be limited but bearable – and covers vectors similar to griefing attacks that can be easily repaired or even gas optimization techniques

### 4.2 Likelihood

- **High** – almost certain to happen and highly lucrative for execution by malicious actors
- **Medium** – still relatively likely, although only conditionally possible incentivize
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

## 5. Audit Summary

The audit duration lasted one week and a total of 21 person days have been spent by the three auditors – @ShieldifyMartin, @ShieldifyAnon and @ShieldifyGhost. The project hasn't undergone a previous security review prior to this one. Overall, we would like to congratulate the team for the amazing idea to develop such a unique project. With some minor exceptions, the code is well-written although there is almost no documentation. Additionally, there is a significant lack of unit tests.

Last but not least, we would like to emphasize that the team has been very communicative and provided detailed answers to all of our questions.

### 5.1 Protocol Summary

| **Project Name** | **GojuoNFT/ZKojuoNFT** |
|---|---|
| **Repository** | N/A |
| **Type of Project** | NFT Collection |
| **Audit Timeline** | 7 days |
| **Review Commit Hash** | N/A |
| **Fixes Review Commit Hash** | N/A |

### 5.2 Scope

The following smart contracts were in the scope of the audit:

| **File** | **nSLOC** |
|---|---|
| src/GojuoNFT.sol | 549 |
| src/MaticGojuoNFT.sol | 248 |
| src/ZKojuoNFT.sol | 296 |
| **Total:** | **1093** |

## 6. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **Critical** and **High** issues: **0**
- **Medium** issues: **0**
- **Low** issues: **3**
- **Informational** issues: **4**
- **Gas Optimizations**issues: **6**

| ID | Title | Severity |
|---|---|---|
| [L-01] | Missing `Re-Entrancy` Protection For `GojuoNFT.claim()` Function | Low |
| [L-02] | Ownership Role Transfer Function Implement Single-Step Role Transfer | Low |
| [L-03] | Missing Zero Address Checks in Many State Changing Functions | Low |
| [I-01] | Missing events affect transparency and monitoring | Informational |
| [I-02] | Unused/Commented Code | Informational |
| [I-03] | Change Function Visibility from `public` to `external` | Informational |
| [I-04] | Missing/Incomplete NatSpec Comments | Informational |
| [G-01] | Using `>>1` Instead of `/2` Can Save Gas | Gas Optimization |
| [G-02] | Splitting `require()` Statements that Use `&&` Saves Gas | Gas Optimization |
| [G-03] | Use string.concat Instead of `abi.encodePacked` | Gas Optimization |
| [G-04] | Use `custom errors` Instead of `require` with Revert Strings | Gas Optimization |
| [G-05] | Array Length Read in Each Iteration of the Loop Wastes Gas | Gas Optimization |
| [G-06] | No Need to Initialize Variables with Default Values | Gas Optimization |

# 7. Findings

## [L-01] Missing `Re-Entrancy` Protection For `GojuoNFT.claim()` Function

### Severity

Low Risk

### Description

A bad actor can perform a `re-entrancy` attack on `GojuoNFT.claim()` function, since they call `safeMint` internally, which calls the `onERC721Received` callback that may contain malicious code.

To protect against cross-function `re-entrancy` attacks, it may be necessary to use a `mutex`. Functions decorated with such modifiers cannot be exploited with recursive calls. OpenZeppelin has its own mutex implementation called `ReentrancyGuard` which provides a `nonReentrant` modifier that guards the decorated function with a mutex against `re-entrancy` attacks.

Also not following the `Checks-Effects-Interactions` pattern, because the `_safeMint()` function is called before the `_tokenMidContents` mapping is updated.

### Location of Affected Code

File: GojuoNFT.sol

```
function claim(
  address _to,
  uint256 _tokenId,
  string memory _tokenMidContent
) public {
.
.
.
  _safeMint(_to, _tokenId);
  _tokenMidContents[_tokenId] = _tokenMidContent;
}
```

### Recommendation

It is recommended to follow the CEI(`Checks-Effects-Interactions`) pattern and use OpenZeppelin's `nonReentrant` modifier for `GojuoNFT.claim()` function.

Example:

```
function claim(
  address _to,
  uint256 _tokenId,
  string memory _tokenMidContent
) external nonReentrant {
.
.
.
- _safeMint(_to, _tokenId);
  _tokenMidContents[_tokenId] = _tokenMidContent;
+ _safeMint(_to, _tokenId);
}
```

### Team Response

Acknowledged

## [L-02] Ownership Role Transfer Function Implement Single-Step Role Transfer

### Severity

Low Risk

### Description

The current ownership transfer process for all the contracts inheriting from `Ownable` involves the current owner calling the `transferOwnership()` function. If the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing access to all functions with the `onlyOwner` modifier.

### Location of Affected Code

`OpenZeppelin Ownable`

```
/**
* @dev Transfers ownership of the contract to a new account (`newOwner`).
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
  require(newOwner != address(0), "Ownable: new owner is the zero address
     ");
  _transferOwnership(newOwner);
}
```

### Recommendation

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an **`acceptOwnership()`** function for the transfer of the ownership to fully succeed. This ensures the nominated EOA account is a valid and active account. This can be easily achieved by using OpenZeppelin's **`Ownable2Step`** contract instead of **`Ownable`**.

```
- import {Ownable} from "@openzeppelin/contracts/access/Ownable.sol";
+ import {Ownable2Step} from "@openzeppelin/contracts/access/Ownable2Step
    .sol";

- contract ZKojuoNFT is ERC721, Ownable {
+ contract ZKojuoNFT is ERC721, Ownable2Step {
```

### Team Response

Acknowledged

## [L-03] Missing Zero Address Checks in Many State Changing Functions

### Severity

Low Risk

### Description

All functions of the smart contracts, are missing address validation. Every input address should be checked not to be zero, especially the ones that could lead to rendering the contract unusable, lock tokens, etc. This is considered a best practice.

### Location of Affected Code

File: ZKojuoNFT.sol

```
function withdrawEth(address _to) public onlyOwner {

function claim(address _to, uint256 _tokenId) public payable {

function claimBatch(address _to, uint256[] memory _tokenIds) public
   payable {

function ownerClaimBatch(
  address _to,
  uint256[] memory _tokenIds
) public onlyOwner {
```

File: GojuoNFT.sol

```
function claim(
  address _to,
  uint256 _tokenId,
  string memory _tokenMidContent
) public {

function ownerClaim(address _to, uint256 _tokenId) public onlyOwner {
```

File: MaticGojuoNFT.sol

```
function ownerClaim(
  address _to,
  uint256 _tokenId
) public onlyOwner onlyValidTokenId(_tokenId) {

function ownerClaimBatch(
  address _to,
  uint256[] memory _tokenIdList
) public onlyOwner {
```

### Recommendation

It is recommended to create a helper function that checks if the address is `address(0)` and use it in all the functions where there is no zero address check.

```
function assemblyOwnerNotZero(address _addr) public pure {
  assembly {
    if iszero(_addr) {
      mstore(0 x00 , "Zero address")
      revert(0 x00 , 0 x20 )
    }
  }
}
```

Note:

Using assembly to check for `address(0)` saves gas.

### Team Response

Acknowledged

## [I-01] Missing events affect transparency and monitoring

### Severity

Informational

### Description

Missing events in critical functions, especially privileged ones, reduce transparency and ease of monitoring. Users may be surprised at changes affected by such functions without being able to observe related events.

### Location of Affected Code

File: GojuoNFT.sol

```
function ownerClaim(address _to, uint256 _tokenId) public onlyOwner {
```

File: MaticGojuoNFT.sol

```
function ownerClaim(
function ownerClaimBatch(
```

File: ZKojuoNFT.sol

```
function claim(address _to, uint256 _tokenId) public payable {
function claimBatch(address _to, uint256[] memory _tokenIds) public
   payable {
function ownerClaimBatch(
function withdrawEth(address _to) public onlyOwner {
function setMintPrice(uint256 _mintPrice) public onlyOwner {
function setRoyaltyPercent(uint256 _royaltyPercent) public onlyOwner {
```

### Recommendation

Consider emitting events in these functions.

### Team Response

Acknowledged

## [I-02] Unused/Commented Code

### Severity

Informational

### Description

There are instances within the code where commented code is left from previous development iterations. While this does not cause any security concerns, it does make the contract less readable for auditors/users.

### Location of Affected Code

File: GojuoNFT.sol

```
// function sliceBytes32To10(bytes32 input) public pure returns (bytes10
   output) {
//   assembly {
//     output := input
//   }
// }
```

File: ZKojuoNFT.sol

```
// import "hardhat/console.sol";

// mapping(address => bool) private gojuonftClassicHolders;

// for (uint256 i = 0; i < _gojuonftClassicHolders.length; i++ ) {
//   gojuonftClassicHolders[_gojuonftClassicHolders[i]] = true;
// }

// if (msg.sender != owner() && gojuonftClassicHolders[msg.sender] ==
   false) {
```

### Recommendation

Consider removing the commented code to improve the readability of the code.

### Team Response

Acknowledged

## [I-03] Change Function Visibility from `public` to `external`

### Severity

Informational

### Description

It is best practice to mark functions that are not called internally as `external` instead, as this saves gas (especially in the case where the function takes arguments, as `external` functions can read arguments directly from `calldata` instead of having to allocate `memory`).

### Location of Affected Code

Most smart contracts.

### Recommendation

Consider changing the visibility of functions that are not used with the contract from `public` to `external`.

### Team Response

Acknowledged

## [I-04] Missing/Incomplete NatSpec Comments

### Severity

Informational

### Description

(`@notice`, `@dev`, `@param` and `@return`) are missing in some functions. Given that NatSpec is an important part of code documentation, this affects code comprehension, audibility, and usability.

This might lead to confusion for other auditors/developers that are interacting with the code.

### Location of Affected Code

All smart contracts.

### Recommendation

Consider adding in full NatSpec comments for all functions where missing to have complete code documentation for future use.

### Team Response

Acknowledged

## [G-01] Using `>>1` Instead of `/2` Can Save Gas

### Severity

Gas Optimization

### Description

A division by `2` can be calculated by shifting one to the right (`>>1`). While the DIV opcode uses `5` `gas`, the `SHR` opcode only uses `3` `gas`.

### Location of Affected Code

Files: MaticGojuoNFT.sol, ZKojuoNFT.sol, GojuoNFT.sol

```
if ((_tokenId / 52) % 2 == 0) {
  shifting_index = _tokenId / 52 / 2;
  prefix = "hiragana-";
} else {
  shifting_index = (_tokenId / 52 - 1) / 2;
  prefix = "katakana-";
}
```

### Recommendation

Consider using shift right for tiny gas optimization.

`/2 -> >> 1`

### Team Response

Acknowledged

## [G-02] Splitting `require()` Statements that Use `&&` Saves Gas

### Severity

Gas Optimization

### Description

Instead of using the `&&` operator in a single `require` statement to check multiple conditions, using multiple require statements with 1 condition per require statement will save 8 GAS per &&. The gas difference would only be realized if the `revert` condition is met.

### Location of Affected Code

All smart contracts.

### Recommendation

Instead of using the && operator in a single require statement to check multiple conditions, use multiple `require` statements with 1 condition per `require` statement.

### Team Response

Acknowledged

## [G-03] Use `string.concat` Instead of `abi.encodePacked`

### Severity

Gas Optimization

### Description

Since solidity version `0.8.12` was released the new method `string.concat` is suggested to be used instead of `abi.encodePacked` for concatenating strings. This is done in multiple places around the current codebase.

### Location of Affected Code

All smart contracts.

### Recommendation

Consider using `string.concat` instead of `abi.encodePacked`.

### Team Response

Acknowledged

## [G-04] Use `custom errors` Instead of `requires` with Revert Strings

### Severity

Gas Optimization

### Description

Custom errors are available from Solidity version `0.8.4`. Custom errors save `~50` gas each time they are hit by avoiding having to allocate and store the revert string. Not defining strings also saves deployment gas.

### Location of Affected Code

All smart contracts.

### Recommendation

Replace all require statements with Solidity **`custom errors`** for better UX and gas savings.

### Team Response

Acknowledged

## [G-05] Array Length Read in Each Iteration of the Loop Wastes Gas

### Severity

Gas Optimization

### Description

Reading array length at each iteration of the loop takes `6` `gas` (3 for mload and 3 to place memory_offset) in the stack. Caching the array length in the stack saves around `3` `gas` per iteration.

### Location of Affected Code

Files: MaticGojuoNFT.sol and ZKojuoNFT.sol

### Recommendation

Cache the array length outside of the loop and use that variable in the loop.

Example:

```
+ uint256 _arrayLength = array.length;

- for (uint256 i; i < array.length; ....) {
+ for (uint256 i; i < _arrayLength; ....) {
```

### Team Response

Acknowledged

## [G-06] No Need to Initialize Variables with Default Values

### Severity

Gas Optimization

### Description

If a variable is not set/initialized, the default value is assumed (0, false, 0x0 ... depending on the data type). Saves 8 gas per instance.

### Location of Affected Code

File: MaticGojuoNFT.sol

```
for (uint256 i = 0; i < _tokenIdList.length; i++) {
```

File: ZKojuoNFT.sol

```
for (uint256 i = 0; i < _tokenIds.length; i++) {
```

### Recommendation

Do not initialize variables with their default values like this:

```
- for (uint256 i = 0; ....
+ for (uint256 i; ....
```

### Team Response

Acknowledged

shieldify

# Thank you!