SMART CONTRACT AUDIT CONCLUSION

by Patrizio Stavola

23rd August 2022

I found no critical bugs, but have discovered some moderate issues.

Findings

ID	Severity	Subject	Status
CVF-1	High	Unused Stata Variable	Info
CVF-2	Minor	Solidity version	Info
CVF-3	Minor	Uninitialized local variable	Info
CVF-4	Minor	Modifier on top	Info
CVF-5	Minor	Use call in place of transfer	Info
CVF-6	Minor	Comments missing	Info

Contents

1 Introduction

The audit goal is a general review of the smart contract structure, critical/major bugs detection and issuing the general recommendations.
The following file was audited:

- src/FirstNft.sol;

1.1 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics. In current audit I use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. i check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. I check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. I also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase I also understand data structures used and the purposes they are used for.

2 Detailed Results

2.1 CVF-1 Unused State Variable

- **Severity** High
- **Status** Info
- **Category** Procedural
- **Source** FirstNft.sol

Description This state variable has been correctly defined but it has never been used.

Listing 1: Unused State Variable

```
15 uint private constant MAX_AMOUNT_PER_TRANSACTION = 5;
```

```
33 require(amount > 0 && amount <= 5, "You can mint at most 5 NFTs in single transaction");
```

2.2 CVF-2 Solidity Version

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** FirstNft.sol

Description Version too recent to be trusted.

Recommendation Consider deploying with 0.8.7

Listing 2: Solidity Version

```
2 pragma solidity ^0.8.13
```

2.3 CVF-3 Uninitialized local variable

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** FirstNft.sol

Recommendation According to common best practice variable should always be initialized with a value even if you are using the default one.

Listing 3: Uninitialized local variable

```
37 for (uint i; i < amount; i++) {
```

2.4 CVF-4 Modifier on top

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** FirstNft.sol

Recommendation According to design pattern you should have modifiers on top, after state variables.

Listing 4: Modifier on top

```
49 modifier onlyOwner() {
```

2.5 CVF-5 Use call in place of transfer

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** FirstNft.sol

Recommendation The current recommended function to send ethers is ‘call’. Check out the sendViaCall example here. One further advantage when using ‘call’ over ‘transfer’ is that it gives you output.

Listing 5: Use call in place of transfer

```
46 payable(msg.sender).transfer(address(this).balance);
```

2.6 CVF-6 Comments missing

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** FirstNft.sol

Recommendation Consider adding mode details about functions and variables purpose and usage. Common best practice is to use natspec comments such as @notice, @params and @devccording.

Listing 6: Comments missing