ABDK CONSULTING

SMART CONTRACT AUDIT

ElephantsLab

OctoGamex NFT Bridge

Solidity

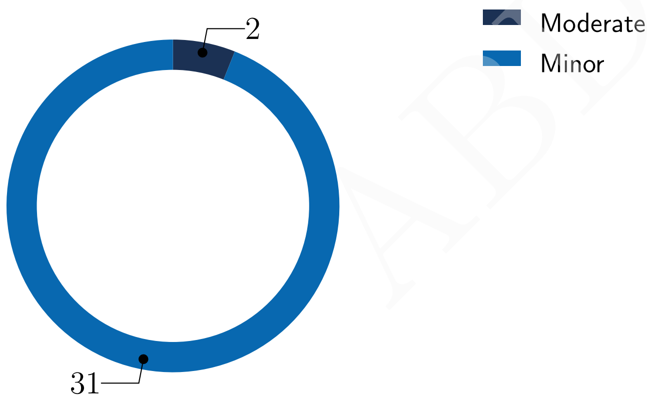
abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich

1st June 2022

We've been asked to review 6 files in a Github repository. We found 2 moderate, and a few less important issues.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Fixed
CVF-2	Minor	Suboptimal	Fixed
CVF-3	Minor	Documentation	Fixed
CVF-4	Minor	Documentation	Fixed
CVF-5	Minor	Bad naming	Info
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Suboptimal	Info
CVF-8	Minor	Suboptimal	Fixed
CVF-9	Moderate	Unclear behavior	Fixed
CVF-10	Minor	Suboptimal	Info
CVF-11	Minor	Suboptimal	Info
CVF-12	Minor	Unclear behavior	Fixed
CVF-13	Minor	Suboptimal	Info
CVF-14	Minor	Bad datatype	Info
CVF-15	Moderate	Suboptimal	Info
CVF-16	Minor	Suboptimal	Info
CVF-17	Minor	Suboptimal	Fixed
CVF-18	Minor	Bad datatype	Info
CVF-19	Minor	Suboptimal	Fixed
CVF-20	Minor	Suboptimal	Fixed
CVF-21	Minor	Suboptimal	Info
CVF-22	Minor	Procedural	Info
CVF-23	Minor	Suboptimal	Info
CVF-24	Minor	Bad datatype	Info
CVF-25	Minor	Procedural	Fixed
CVF-26	Minor	Bad datatype	Info
CVF-27	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-28	Minor	Bad datatype	Info
CVF-29	Minor	Procedural	Fixed
CVF-30	Minor	Bad datatype	Info
CVF-31	Minor	Procedural	Info
CVF-32	Minor	Suboptimal	Info
CVF-33	Minor	Bad datatype	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	June 1, 2022	D. Khovratovich	Initial Draft
0.2	June 1, 2022	D. Khovratovich	Minor revision
1.0	June 1, 2022	D. Khovratovich	Release
1.1	June 1, 2022	D. Khovratovich	Cover page fix
2.0	June 1, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the contracts at repository:

- interfaces/INFTToken.sol
- tokens/MultiNFTToken.sol
- tokens/NFTToken.sol
- tokens/SimpleMultiNFT.sol
- tokens/SimpleNFT.sol
- NFTBridge.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we 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. We 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.

- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.

- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We 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 we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** NFTBridge.sol

Recommendation Should be "^0.8.0" as major releases may break backward compatibility. Also relevant for the next files: SimpleMultiNFT.sol, NFTToken.sol, MultiNFTToken.sol, SimpleNFT.sol, INFTToken.sol.

Listing 1:

```
3 pragma solidity >=0.8.0;
```

3.2 CVF-2

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** NFTBridge.sol

Description Declaring a top-level structure in a file named after a contract makes it harder to find the structure in the code.
Recommendation Consider either moving the structure declaration into the contract or moving it into a separate file named "types.sol" or something line this.

Listing 2:

```
9 struct Lock {
```

3.3 CVF-3

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** NFTBridge.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider documenting.

Listing 3:

```
24 mapping(bytes32 => mapping(uint256 => bool)) private
     ↪ receivedClaims;
```

3.4 CVF-4

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** NFTBridge.sol

Description The semantics of the keys and values in this mapping is unclear.
Recommendation Consider documenting.

Listing 4:

```
mapping(bytes32 => mapping(address => address)) public
  ↪ bridgedTokens;
```

3.5 CVF-5

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** NFTBridge.sol

Recommendation Events are usually named via nouns, such as "Lock" and "Claim".
Client Comment Decided not to fix. We have struct with the same name (Lock) already and we have already developed events scanner with such names.

Listing 5:

```
event Locked(

event Claimed(
```

3.6 CVF-6

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

Description This parameter is always equal to the block.timestamp, which can be retrieved for free from the event itself.
Recommendation Consider dropping this parameter.
Client Comment Decided not to fix. We have had some problems with block timestamp receiving in some chains before. So decided to force this value in the events directly.

Listing 6:

```
uint256 timestamp

uint256 timestamp

uint256 timestamp
```

3.7 CVF-7

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

Recommendation This check is redundant as it is anyway possible to pass dead fee reciever address.
Client Comment Decided not to fix.

Listing 7:

```
require(feeReceiver_ != address(0x0), "Invalid fee receiver
  ↪ address");
```

3.8 CVF-8

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** NFTBridge.sol

Recommendation The interface ID used here could be obtained as " type(INFTToken).interfaceId".

Listing 8:

```
if (NFTToken(nftContractAddress).supportsInterface(bytes4(
  ↪ keccak256('bridgeAddress()')))) {

if (MultiNFTToken(nftContractAddress).supportsInterface(bytes4(
  ↪ keccak256('bridgeAddress()')))) {

 && NFTToken(bridgedTokens[source][nftContractAddress]).
   ↪ supportsInterface(bytes4(keccak256('bridgeAddress()'))))

 && MultiNFTToken(bridgedTokens[source][nftContractAddress]).
   ↪ supportsInterface(bytes4(keccak256('bridgeAddress()'))))
```

3.9 CVF-9

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** NFTBridge.sol

Description It is not enough to check that a token implements the "INFTToken" interface, as there could be different bridges and the token may belong to another bridge.
Recommendation Consider also checking that nftContractAddress.bridgeAddress() == this.

Listing 9:

```
if (NFTToken(nftContractAddress).supportsInterface(bytes4(keccak256('bridgeAddress()')))) {

if (MultiNFTToken(nftContractAddress).supportsInterface(bytes4(keccak256('bridgeAddress()')))) {

 && NFTToken(bridgedTokens[source][nftContractAddress]).supportsInterface(bytes4(keccak256('bridgeAddress()'))))

 && MultiNFTToken(bridgedTokens[source][nftContractAddress]).supportsInterface(bytes4(keccak256('bridgeAddress()'))))
```

3.10 CVF-10

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

Description The expression "Address.isContract(bridgedTokens[source][nftContractAddress])" is calcualted several times.
Recommendation Consider calculating once and reusing.
Client Comment Decided not to fix. Stack too deep, try removing local variables error occured.

Listing 10:

```
require(Address.isContract(bridgedTokens[source][nftContractAddress]), "The associated token didn't register");

if (Address.isContract(bridgedTokens[source][nftContractAddress]))

if (Address.isContract(bridgedTokens[source][nftContractAddress]))
```

3.11 CVF-11

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

Description The expression "bridgedTokens[source][nftContractAddress]" is calcualted several times.
Recommendation Consider calculating once and reusing.
Client Comment Decided not to fix. Stack too deep, try removing local variables error occured.

Listing 11:

```
120    require(Address.isContract(bridgedTokens[source][
         ↪ nftContractAddress]), "The associated token didn't
         ↪ register");

128    if (Address.isContract(bridgedTokens[source][
         ↪ nftContractAddress])
     && NFTToken(bridgedTokens[source][nftContractAddress]).
         ↪ supportsInterface(bytes4(keccak256('bridgeAddress()')))
         ↪ )

131      NFTToken(bridgedTokens[source][nftContractAddress]).mint(
           ↪ recipient, tokenId);

136    if (Address.isContract(bridgedTokens[source][
         ↪ nftContractAddress])
     && MultiNFTToken(bridgedTokens[source][nftContractAddress]).
         ↪ supportsInterface(bytes4(keccak256('bridgeAddress()')))
         ↪ )

139      MultiNFTToken(bridgedTokens[source][nftContractAddress]).
           ↪ mint(recipient, tokenId, amount, hex"00");

145  emit Claimed(nftContractAddress, bridgedTokens[source][
       ↪ nftContractAddress], recipient, tokenId, amount, lockIdx,
       ↪ source, block.timestamp);
```

3.12 CVF-12

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** NFTBridge.sol

Description These functions should emit some events.

Listing 12:

```
function changeOracle(address _oracle) external onlyOwner {

function setFee(uint256 fee_) external onlyOwner {

function changeFeeReceiver(address feeReceiver_) external
    ↪ onlyOwner {
```

3.13 CVF-13

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

Description This check is redundant, as it is anyway possible to pass a dead fee receiver address.
Client Comment Decidede not to fix.

Listing 13:

```
require(feeReceiver_ != address(0x0), "Invalid fee receiver
    ↪ address");
```

3.14 CVF-14

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** NFTBridge.sol

Recommendation The return type should be "NFTToken" or some interface extracted from it.
Client Comment Decidede not to fix. We use this return for the automatici tests. Why can't it be an address?

Listing 14:

```
) external returns(address) {
```

3.15 CVF-15

- **Severity** Moderate
- **Category** Suboptimal
- **Status** Info
- **Source** NFTBridge.sol

Description The values been hashed are concatenated without separators, thus value boundaries are not preserved. If one token has name="foo" and symbol="bar" and another token has name="foob" and symbol="ar", then hashes will be the same.
Recommendation Consider either using the "abi.encode" function or hashing recursively or using some other approach that preserves value boundaries.
Client Comment Decided not to fix. These messages are signed with the Oracle and even if tokens will have such problem, they will have edifferent contract addresses.

Listing 15:

```
187 bytes32 hash = keccak256(abi.encodePacked(name, symbol, uri,
    ↪ source, originalNFTContractAddress, destination));

219 bytes32 hash = keccak256(abi.encodePacked(uri, source,
    ↪ originalNFTContractAddress, destination));
```

3.16 CVF-16

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

Description Deploying a new contract instance every time is suboptimal.
Recommendation Consider deploying a minimal proxy as described in EIP-1167.
Client Comment Decided not to fix.

Listing 16:

```
190 address nftContractAddress = address(new NFTToken(name, symbol,
    ↪ uri, originalNFTContractAddress, address(this)));

222 address nftContractAddress = address(new MultiNFTToken(uri,
    ↪ originalNFTContractAddress, address(this)));
```

3.17 CVF-17

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** NFTBridge.sol

Description Strings are inefficient.
Recommendation Consider using enum constants instead.

Listing 17:

```
198 "ERC721",

230 "ERC1155",
```

3.18 CVF-18

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** NFTBridge.sol

Recommendation The return type should be "MultiNFTToken" or some interface extracted from it.
Client Comment Decidede not to fix. We use this return for the automatici tests. Why can't it be an address?

Listing 18:

```
211 ) external returns(address) {
```

3.19 CVF-19

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** NFTBridge.sol

Recommendation This line could be simplified as: v := mload(add(sig, 65))

Listing 19:

```
258 v := byte(0, mload(add(sig, 96))) // final byte (first byte of
      ↪ the next 32 bytes)
```

3.20 CVF-20

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** NFTBridge.sol

Description This line effectively does nothing .
Recommendation Consider removing it.

Listing 20:

```
261 return (v, r, s);
```

3.21 CVF-21

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

Description In case of invalid signature, zero is silently returned here.
Recommendation Consider reverting in such a case.
Client Comment Decided not to fix.

Listing 21:

```
267 return ecrecover(message, v, r, s);
```

3.22 CVF-22

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** SimpleMultiNFT.sol

Recommendation This contract looks like a test stuff, it shouldn't be in the production code base.
Client Comment Decided not to fix.

Listing 22:

```
7 contract SimpleMultiNFT is ERC1155("test") {
```

3.23 CVF-23

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

Recommendation The number of different tokens to mint and the amount of each token should be constructor arguments.
Client Comment Decided not to fix.

Listing 23:

```
for (uint8 i = 0; i < 10; i++) {
  _mint(msg.sender, i, 100, "");
```

3.24 CVF-24

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** NFTToken.sol

Recommendation The type of this variable should be more specific.
Client Comment Decided not to fix.

Listing 24:

```
address public original;
```

3.25 CVF-25

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** NFTToken.sol

Recommendation These variables should be declared as immutable.

Listing 25:

```
address public original;
address public bridgeAddress;
```

3.26 CVF-26

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** NFTToken.sol

Recommendation The type of this argument should be more specific.
Client Comment Decided not to fix.

Listing 26:

```
address original_ ,
```

3.27 CVF-27

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

Description This function wouldn't be necessary if the "baseURI" variable would be declared as internal.
Client Comment Decided not to fix. We should reload this method from the ERC721 lib.

Listing 27:

```
function _baseURI() internal view override returns (string
    ↪ memory) {
```

3.28 CVF-28

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** MultiNFTToken.sol

Recommendation The type of this variable should be more specific.
Client Comment Decided not to fix.

Listing 28:

```
address public original;
```

3.29 CVF-29

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** MultiNFTToken.sol

Recommendation These variables should be declared as immutable.

Listing 29:

```
address public original;
address public bridgeAddress;
```

3.30 CVF-30

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** MultiNFTToken.sol

Recommendation The type of the "original_" argument should be more specific.
Client Comment Decided not to fix.

Listing 30:

```
constructor(string memory uri_, address original_, address
    ↪ bridgeAddress_) ERC1155(uri_) {
```

3.31 CVF-31

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** SimpleNFT.sol

Recommendation This contract looks like a test stuff, it shouldn't be in the production code base.
Client Comment Decided not to fix.

Listing 31:

```
contract SimpleNFT is ERC721("NFTS", "Simple NFT") {
```

3.32 CVF-32

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

Recommendation The number of NFT to mint should be a constructor argument.
Client Comment Decided not to fix.

Listing 32:

```
for (uint8 i = 0; i < 10; i++) {
```

3.33 CVF-33

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** INFTToken.sol

Recommendation The return type should be more specific. Like an interface extracted from the "NFTBridge" contract.
Client Comment Decided not to fix.

Listing 33:

```
function bridgeAddress() external view returns (address);
```