True NFT informal audit report

Prepared by Pruvendo
09/24/21

Project summary

The present audit report provides the findings about the current version of True NFT smart contract jointly developed by RSquad and TonLabs. While the auditors did their best to discover all the defects, the present activity is just an initial step for the incoming stages of the full-scale formal verification. Consequently, the positive result of the present audit should not be considered as any kind of ultimate statement in terms of the quality of the present contract. However, the positive conclusion can encourage the business to deploy the preliminary version of the contract into limited production keeping, at the same time, all the responsibility for the hidden defects of the contract that still did not pass the full-scale formal verification.

The audit was run against *true-nft-core* hashed d7dabed756d28582450ea2e7d2a964fbd19d4c29, all the issues were shared with the authors using NFT Pruvendo+RSquad+TON Labs Telegram group in a timely manner.

Any questions related to the present audit can be addressed to its author via Telegram @SergeyEgorovSPb .

Brief contract summary

Business-level Summary

The present contract represents the paradigm of NFT - the unique token that represents some data (for example, picture, music clip or just a hash number) with a clear ownership. The only operation that ought to be supported is transferring ownership to another account. Also it's important to obtain the list of available NFT tokens to select one desirable for purchase.

Description of system

The system is built on the concept that blockchain is `Key → Value` storage. It consists of four smart contracts, two of which are implemented by NFT, and two are required for fast searches in the blockchain.
It is important to note, the data that make up the NFT content is deployed once and does not affect the transfer of ownership. At the same time, you can easily find all NFT collections by the codeHash of the IndexBasis and Index contracts, while the code of these contracts is assumed to be unchanged for all NFT implementations. This approach will provide a global search across all NFT implementations
Based on this, it becomes obvious that the problem of storing data in ledgers or otherwise has been solved.

Developer-level brief description

Contracts

`NftRoot` - A smart contract that is responsible for the release of NFT.

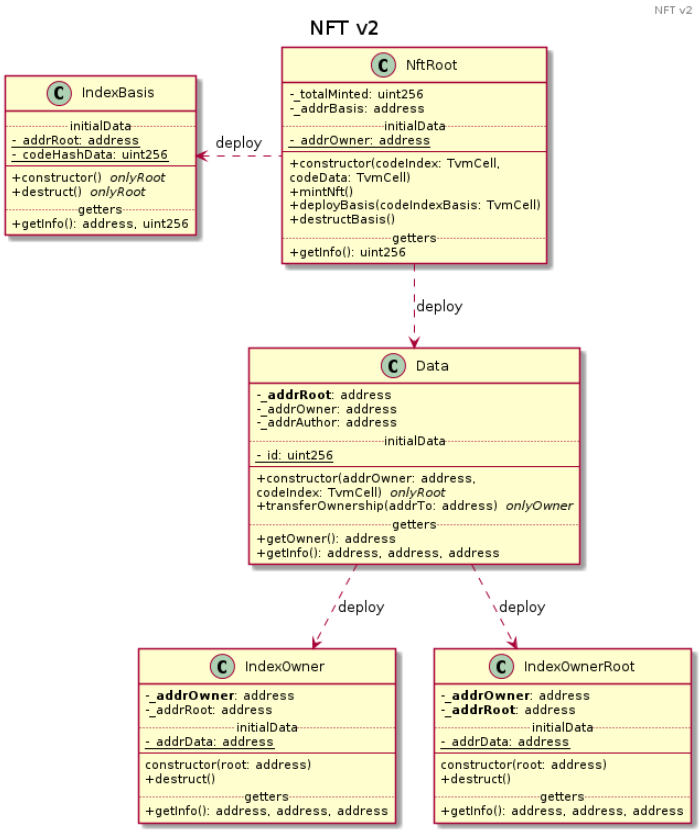
`Data` - The contract stores information, which is essentially NFT, and is also responsible for changing the owner.

`Index` - The contract that is used to find all NFTs for a specific owner.

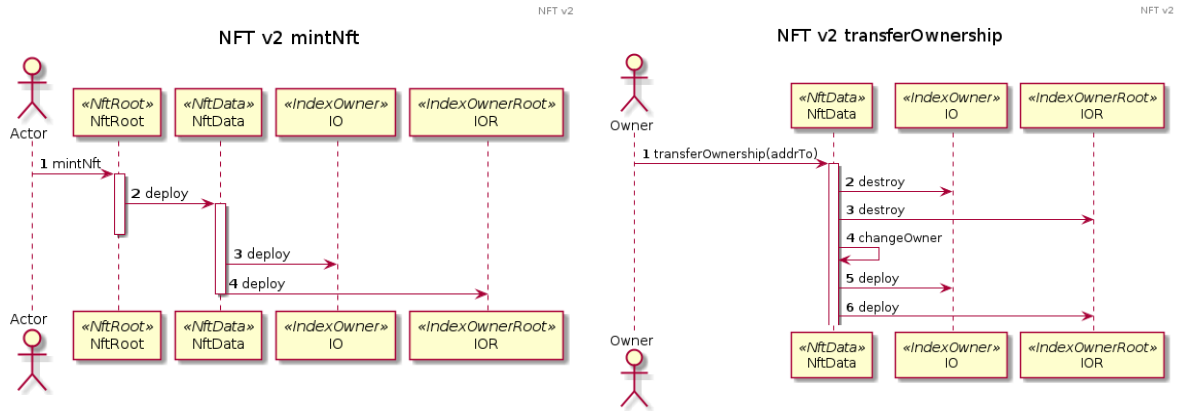
`IndexBasis` - The contract that is used to find all Roots and all NFTs.

It's important to mention that two former contracts are customizable and the current versions are created mostly for development purposes. It's why for the present report all the issues related to these contracts never have a severity higher than MINOR.

Class diagram



Transfer sequence diagram



Logic and Data

`NftRoot` and `Data` separate data storage and logic in the system. `Data` is a data warehouse that can contain content of a completely different nature, up to images. He is responsible for the transfer of ownership. `NftRoot` is responsible for minting NFT, but may contain other functionality such as writing.

Search

The global search is carried out using the `IndexBasis` contract, which contains the `NftRoot` address. Since the `NftRoot` address is included in the initialData of IndexBasis, there can be one basis for one root. It also contains the `Data codeHash`, which in turn is salted with the root address. Using this `codeHash`, you can find all the `Data` for this root. Using the `Index` contract code, you can find all your NFT with one query.

Audit results

Brief methodology description

The audit contains three parts: functional-level audit , cross-functional audit and non-functional one . The main goal of the former one is to check that each function works correctly while the second is mostly related to the correctness of the interactions between different functions and the last one checks such stuff like global variables, constants etc.

It's important to mention that here functions are considered exclusively as handlers of messages so any kind of inline, private, modifiers or off-chain functions are audited together with their callers and not considered as a separate entity.

The checklist for the functional level looks as follows:

- Access check
- Check for malicious data handling
- Check for correctness of parameters/global state and error handling
- Correctness of business logic
- Correctness of branches and loops
- Check for reasonable gas usage
- Check for correctness of all the transfers, including change handling
- Check for limited amount of message calls (need to be less than 255)

The checklist for cross-functional level looks as follows:

- No unexpected calls
- No moving to incorrect state due to lack of funding for inner calls
- No access violation through the outer methods
- No stall in the intermediate state

The checklist for non-functional level is as follows:

- Check initial values for global variables
- Check values for constants
- Check violation of coding style, naming conventions etc.

Functional-level audit

Contract `NftRoot`

IMPORTANT NOTE!!! As this contract is customizable and was developed mostly for demonstration purposes so all the discovered issues are considered as MINOR.

constructor

The method is called upon creation of the `NftRoot` contract.

Checklist:

Check type	Result
Access	OK
Malicious data handling	Malicious cells for data and index can be sent by the evil-minded creator. Severity : MINOR Suggested fix : it's a well-known architectural issue, without any known good solution. No suggested fixes are provided.

Input correctness and error handling	Providing zero as a parameter value will lead to poor predictable effects
	
	Severity : MINOR
	
	Suggested fix : insert
	`require (codeIndex != 0);`
	`require (codeData !=0);`
	
	before `tvm.accept();`

Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	N/A

mintNft

The method creates a new NFT token.

Checklist:

Check type	Result
Access	Anyone can call this method while only the owner should have such a privilege. Severity : MINOR Suggested fix : a) introduce `_ownerAddr` variable assigned in the constructor b) insert `require (msg.sender == owner.addr);` in the beginning of the function
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	OK
Other	The new `Data` contract gets 1.1 ton It's suggested to make 1.1 ton as a constant derived from `MIN_FOR_DEPLOY`

deployBasis

This method deploys a new `IndexBasis` contract.

Checklist:

Check type	Result
Access	Anyone can call this method while only the owner should have such a privilege. Severity : MINOR Suggested fix : c) introduce `_ownerAddr` variable assigned in the constructor d) insert `require (msg.sender == owner.addr);` in the beginning of the function
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	OK
Other	The variable named `codeHasData` has a typo

destroyBasis

This method destroys the `IndexBasis`.

Checklist:

Check type	Result
Access	Anyone can call this method while only the owner should have such a privilege. Severity : MINOR Suggested fix : e) introduce `_ownerAddr` variable assigned in the constructor f) insert `require (msg.sender == owner.addr);` in the beginning of the function
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	OK

Contract `Data`

IMPORTANT NOTE!!! As this contract is customizable and was developed mostly for demonstration purposes so all the discovered issues are considered as MINOR.

constructor

Creates and initializes a new `Data` contract.

Checklist:

Check type	Result
Access	OK
Malicious data handling	OK
Input correctness and error handling	An external sender can encode an index cell and publish a new Data contract with the “zero” owner. Looks like it doesn’t open any immediate possibilities for an attack but definitely contradicts the desired logic. Severity : MINOR Suggested fix : introduce `require (msg.sender != address(0));`
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	OK

transferOwnership

The method transfers ownership of NFT token from one owner to another.

Checklist:

Check type	Result
Access	OK
Malicious data handling	OK
Input correctness and error handling	Setting the `addrTo` input parameter to zero can move the NFT token to “public domain”. While it is not an important problem it definitely looks like undesirable behavior. Severity : MINOR Suggested fix : introduce `require (addrTo != address(0));`
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change stealing it. Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	OK

getInfo

Getter that provides generic information about the `Data` contract.

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	N/A
Transfers	N/A
Messages	N/A

getOwner

Getter that provides generic information about the owner of the Data contract.

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	N/A
Transfers	N/A
Messages	N/A

Contract `index`

constructor

Creates and initializes a new `Index` contract.

Checklist:

Check type	Result
Access	OK
Malicious data handling	OK
Input correctness and error handling	An external sender can encode an index cell and publish a new `Index` contract with the “zero” owner. Looks like it doesn’t open any immediate possibilities for an attack but definitely contradicts the desired logic. Severity : MINOR Suggested fix : introduce `require (msg.sender != address(0));`
	Setting the `addrTo` input parameter to zero can make a poorly predictable index for “public” NFTs. It seems it doesn’t open any possibilities to attack but definitely contradicts the desired logic. Severity : MINOR Suggested fix : introduce `require (addrTo != address(0));`
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change, keeping it as a deposit throughout the whole lifecycle of the index .

	Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	N/A

getInfo

Returns information about index

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	N/A
Transfers	N/A
Messages	N/A

destruct

Destructs an index

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	OK
Messages	N/A

Contract `IndexBasis`

constructor

Creates and initializes a new `IndexBasis` contract

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	The method does not return change, keeping it as a deposit throughout the whole lifecycle of the index . Severity : MINOR Suggested fix : insert `msg.sender.transfer(0, false, 64);` at the end of the message
Messages	N/A

getInfo

Returns information about the index basis

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	N/A
Transfers	N/A
Messages	N/A

destruct

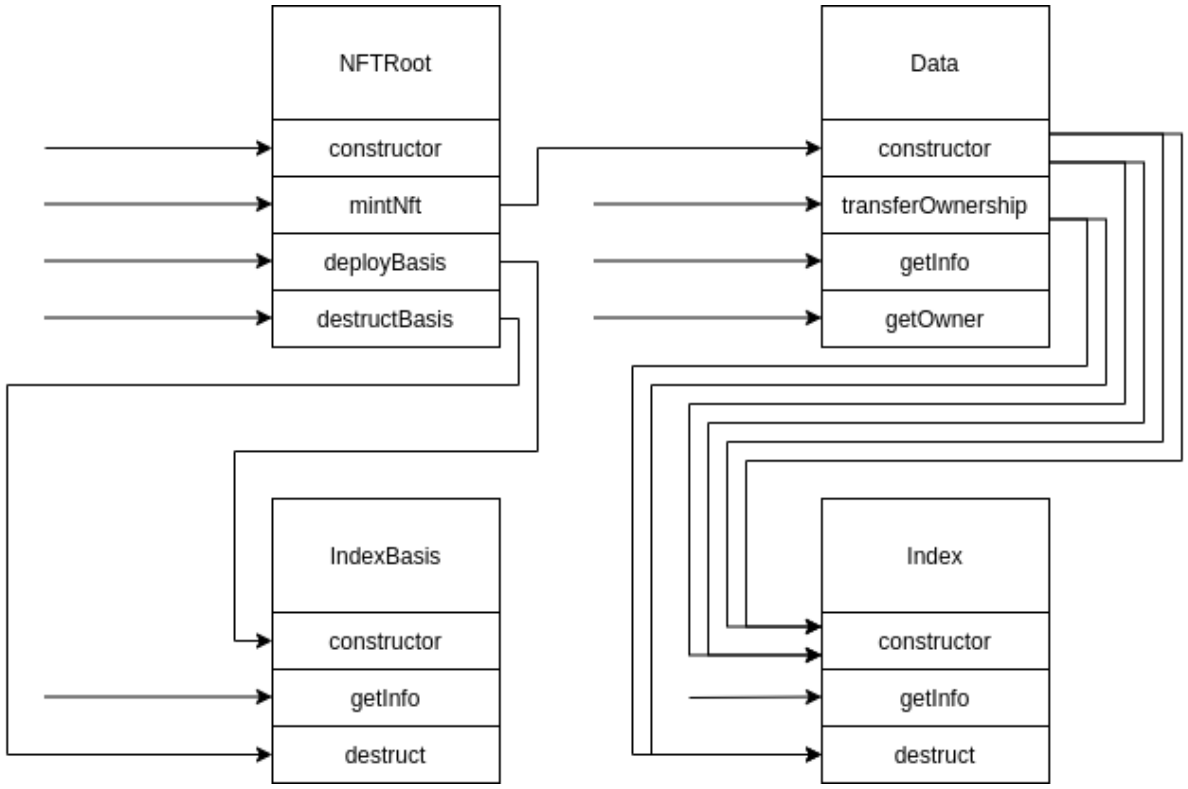
Destructs the index basis

Checklist:

Check type	Result
Access	OK
Malicious data handling	N/A
Input correctness and error handling	N/A
Business logic	OK
Branches and loops	N/A
Gas usage	OK
Transfers	OK
Messages	N/A

Cross-functional level audit

Call tree



Cross-functional checklist

Note: any issues duplicated to the ones found at the functional-level audit are not included here.

Item	Result
No unexpected calls	OK
No moving to incorrect state due to lack of funding for inner calls	OK
No access violation through the outer methods	`IndexBasis` can be destructed by anybody who has an access to `destructBasis` of `NftRoot`. It's clear that a customized version of `NftRoot` will

	set some access restrictions on this method, however the existence of such a method clearly indicates that at least somebody (presumably, the initial creator of the `NftRoot`) will have access to it. Thus, at any moment somebody is able to remove the tokens from the indices preventing the honest token owners from the usage of regular trading services.
No stall in the intermediate state	OK

Non-functional level audit

Item	Results
State variables	OK
Constants	OK
Modifiers	OK

Audit summary

Contract	Function	Severity	Description
`IndexBasis` `/NftRoot`	`destruct/` `destructBasis`	CRITICAL	`IndexBasis` can be destructed by anybody who has an access to `destructBasis` of `NftRoot`. It's clear that a customized version of `NftRoot` will set some access restrictions on this method, however the existence of such a method clearly indicates that at least somebody (presumably, the initial creator of the `NftRoot`) will have access to it. Thus, at any moment somebody is able to remove the tokens from the indices preventing the honest token owners from

			the usage of regular trading services.
NftRoot	constructor	MINOR	Malicious input parameters can be sent to the constructor.
			Providing zero as a parameter value will lead to poor predictable effects
	constructor		The method does not return change stealing it.
	mintNft		
	deployBasis		
	destroyBasis		
	mintNft		Anyone can call this method while only the owner should have such a privilege
	deployBasis		
	destroyBasis		
Data	constructor		An external sender can encode an index cell and publish a new Data contract with the “zero” owner. Looks like it doesn’t open any immediate possibilities for an attack but definitely contradicts the desired logic.
	transferOwnership		Setting the addrTo input parameter to zero can move the NFT token to “public domain”. While it is not an important problem it definitely looks like undesirable behavior.
	constructor		The method does not return change stealing it.
	transferOwnership		
Index	constructor		An external sender can encode an index cell and publish a new Index contract with the “zero” owner. Looks like it doesn’t open any immediate possibilities for an attack but definitely contradicts the desired logic.
			Setting the addrTo input parameter to zero can make a poorly predictable index for “public” NFTs. It seems it doesn’t open any possibilities to attack but definitely contradicts the desired logic.
			The method does not return change, keeping it as a deposit throughout the

`IndexBasis`	`constructor`		whole lifecycle of the index .
`NftRoot`	`mintNft`	NOTE	The new `Data` contract gets 1.1 ton It's suggested to make 1.1 ton as a constant derived from `MIN_FOR_DEPLOY`
	`deployBasis`		The variable named `codeHasData` has a typo

Final conclusion

The present audit confirmed the high quality of the code being reviewed. However, before moving to the next stages of the formal verification it's :

- Necessary to address the only critical issue found
- Highly recommended to address the minor issues to ensure the best practices are advised for the developers of the customized versions