ABDK CONSULTING

SMART CONTRACT AUDIT

ZKSwap

V3

Solidity

abdk.consulting

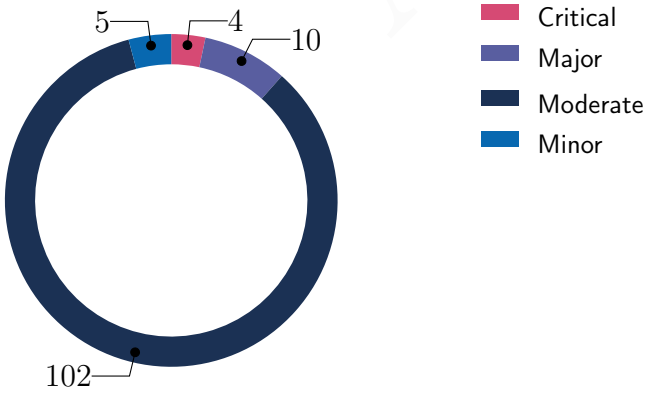
SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
13th April 2022

We've been asked to review files in two ZKSwap repositories, which together comprise ZkSwap V3 version:

- zkswap-v2.
- zkswap-v3.

We found 4 critical, 10 major, and a few less important issues.
All identified critical issues have been fixed. All identified major issues have been fixed or otherwise addressed in collaboration with the client.



Findings

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

ID	Severity	Category	Status
CVF-28	Minor	Procedural	Info
CVF-29	Major	Unclear behavior	Info
CVF-30	Moderate	Unclear behavior	Info
CVF-31	Minor	Readability	Info
CVF-32	Minor	Procedural	Info
CVF-33	Minor	Documentation	Info
CVF-34	Minor	Documentation	Info
CVF-35	Minor	Documentation	Info
CVF-36	Major	Flaw	Fixed
CVF-37	Minor	Bad datatype	Info
CVF-38	Minor	Bad naming	Info
CVF-39	Minor	Unclear behavior	Info
CVF-40	Minor	Procedural	Info
CVF-41	Minor	Procedural	Info
CVF-42	Major	Suboptimal	Info
CVF-43	Minor	Suboptimal	Info
CVF-44	Minor	Bad datatype	Info
CVF-45	Minor	Bad datatype	Info
CVF-46	Minor	Documentation	Info
CVF-47	Minor	Procedural	Info
CVF-48	Minor	Bad datatype	Info
CVF-49	Minor	Bad naming	Info
CVF-50	Minor	Flaw	Info
CVF-51	Minor	Bad naming	Info
CVF-52	Minor	Suboptimal	Info
CVF-53	Minor	Suboptimal	Info
CVF-54	Major	Suboptimal	Info
CVF-55	Moderate	Suboptimal	Info
CVF-56	Minor	Procedural	Info
CVF-57	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-58	Minor	Procedural	Info
CVF-59	Minor	Suboptimal	Info
CVF-60	Minor	Suboptimal	Info
CVF-61	Minor	Bad datatype	Info
CVF-62	Minor	Suboptimal	Info
CVF-63	Minor	Overflow/Underflow	Info
CVF-64	Minor	Bad datatype	Info
CVF-65	Minor	Overflow/Underflow	Info
CVF-66	Major	Unclear behavior	Fixed
CVF-67	Minor	Flaw	Info
CVF-68	Minor	Procedural	Info
CVF-69	Minor	Procedural	Info
CVF-70	Minor	Unclear behavior	Info
CVF-71	Major	Suboptimal	Info
CVF-72	Minor	Procedural	Info
CVF-73	Minor	Unclear behavior	Info
CVF-74	Minor	Unclear behavior	Info
CVF-75	Minor	Unclear behavior	Info
CVF-76	Minor	Procedural	Info
CVF-77	Minor	Suboptimal	Info
CVF-78	Minor	Suboptimal	Info
CVF-79	Minor	Suboptimal	Info
CVF-80	Minor	Suboptimal	Info
CVF-81	Moderate	Flaw	Info
CVF-82	Minor	Suboptimal	Info
CVF-83	Minor	Procedural	Info
CVF-84	Minor	Procedural	Info
CVF-85	Minor	Suboptimal	Info
CVF-86	Minor	Procedural	Info
CVF-87	Minor	Documentation	Info

ID	Severity	Category	Status
CVF-88	Minor	Bad naming	Info
CVF-89	Minor	Suboptimal	Info
CVF-90	Minor	Bad naming	Info
CVF-91	Minor	Suboptimal	Info
CVF-92	Minor	Suboptimal	Info
CVF-93	Minor	Documentation	Info
CVF-94	Minor	Documentation	Info
CVF-95	Minor	Bad datatype	Info
CVF-96	Major	Flaw	Fixed
CVF-97	Minor	Unclear behavior	Info
CVF-98	Minor	Documentation	Info
CVF-99	Minor	Overflow/Underflow	Info
CVF-100	Minor	Bad datatype	Info
CVF-101	Minor	Readability	Info
CVF-102	Minor	Bad datatype	Info
CVF-103	Major	Procedural	Fixed
CVF-104	Minor	Procedural	Info
CVF-105	Minor	Documentation	Info
CVF-106	Minor	Documentation	Info
CVF-107	Minor	Documentation	Info
CVF-108	Minor	Procedural	Info
CVF-109	Minor	Suboptimal	Info
CVF-110	Minor	Suboptimal	Info
CVF-111	Minor	Suboptimal	Info
CVF-112	Minor	Suboptimal	Info
CVF-113	Minor	Bad naming	Info
CVF-114	Minor	Suboptimal	Info
CVF-115	Minor	Suboptimal	Info
CVF-116	Minor	Suboptimal	Info
CVF-117	Moderate	Flaw	Info

ID	Severity	Category	Status
CVF-118	Minor	Documentation	Info
CVF-119	Minor	Bad naming	Info
CVF-120	Minor	Procedural	Info
CVF-121	Major	Flaw	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	April 13, 2022	D. Khovratovich	Initial Draft
0.2	April 13, 2022	D. Khovratovich	Minor revision
1.0	April 13, 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.
A difference between the next commits was the audit subject:

- commit 72ffa2d
- commit 1cc5b18

We have reviewed the next files:

- witness/mod.rs
- witness/nft_approve.rs
- witness/nft_deposit.rs
- witness/nft_exchange.rs
- witness/nft_full_exit.rs
- witness/nft_mint.rs
- witness/nft_transfer_to_new.rs
- witness/nft_transfer.rs
- witness/nft_withdraw.rs
- witness/utils.rs
- witness/add_liquidity.rs
- witness/change_pubkey_offchain.rs
- witness/close_account.rs
- witness/deposit.rs
- witness/full_exit.rs
- witness/remove_liquidity.rs
- witness/swap.rs
- witness/tests/mod.rs
- witness/transfer.rs
- witness/transfer_to_new.rs
- witness/withdraw.rs

- account.rs
- element.rs
- nft_exit_circuit.rs
- exit_circuit.rs
- lp_exit_circuit.rs
- nft/libs/Address.sol
- nft/libs/EnumerableMap.sol
- nft/libs/EnumerableSet.sol
- nft/libs/ERC721.sol
- nft/libs/IERC165.sol
- nft/libs/IERC721.sol
- nft/libs/IERC721Enumerable.sol
- nft/libs/IERC721Metadata.sol
- nft/libs/IERC721Receiver.sol
- nft/libs/Strings.sol
- nft/IZKBoxNFT.sol
- nft/OwnableContract.sol
- nft/ZKBoxNFT.sol
- uniswap/UniswapV2ERC20.sol
- Bytes.sol
- VerifierExit.sol
- ZkBox.sol
- ZkSyncExit.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** Info
- **Category** Suboptimal
- **Source** EnumerableMap.sol

Description This structure requires two storage reads in order to obtain the value by a key.
Recommendation Consider storing only the keys in the array, and value+index pairs in the mapping to make mapping access more efficient.

Listing 1:

```
46 +MapEntry[] _entries;

50 +mapping (bytes32 => uint256) _indexes;
```

3.2 CVF-2

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

Recommendation This should be done only when toDeleteIndex != lastIndex.

Listing 2:

```
96 +MapEntry storage lastEntry = map._entries[lastIndex];

99 +map._entries[toDeleteIndex] = lastEntry;

101 +map._indexes[lastEntry._key] = toDeleteIndex + 1; // All
    ↪ indexes are 1-based
```

3.3 CVF-3

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

Recommendation The term "size" is more common for maps than "length".

Listing 3:

```
125 +function _length(Map storage map) private view returns (uint256
    ↪ ) {
```

3.4 CVF-4

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

Description the semantics of the returned values is unclear.
Recommendation Consider documenting.

Listing 4:

```
139 +function _at(Map storage map, uint256 index) private view
    ↪ returns (bytes32, bytes32) {

150 +function _tryGet(Map storage map, bytes32 key) private view
    ↪ returns (bool, bytes32) {
```

3.5 CVF-5

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

Description This check is redundant, as Solidity compiler will do the same check at the next line.

Listing 5:

```
140 +require(map._entries.length > index, "EnumerableMap: index out
    ↪ of bounds");
```

3.6 CVF-6

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

Description This index is off by 1 from the index in ‘_indexes‘. This is error prone.

Listing 6:

```
142 +MapEntry storage entry = map._entries[index];
```

3.7 CVF-7

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

Description There seems to be only one place where these functions are actually used.
Recommendation Consider rewriting that code to use "tryGet" and removing these functions.

Listing 7:

```
172 +* CAUTION: This function is deprecated because it requires
    ↪ allocating memory for the error

260 +* CAUTION: This function is deprecated because it requires
    ↪ allocating memory for the error
```

3.8 CVF-8

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** EnumerableMap.sol

Description For an enumerable map whose values are addresses a more efficient structure is possible.
Recommendation Such map could be represented as the following two collections: 1. An array of keys 2. A mapping from a key to a 32-bytes word containing the value (address) and the index of the key inside the array As an address occupies only 20 bytes, 12 bytes remain for an index, which allows maps of up to about 8e28 elements.

Listing 8:

```
183 +struct UintToAddressMap {
+     Map _inner;
+}
```

3.9 CVF-9

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Description This condition is not properly documented.
Recommendation Consider referring to some documentation where this case is described

Listing 9:

```
782 +let amm_is_left = multi_or(
+     cs.namespace(|| "amm_is_left"),
+     &[is_zero.clone(), is_two.clone(), is_four.clone()],
+)?;
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Description Cloning "is_zero" and "is_four" here is redundant, as these variables are not used below.

Listing 10:

```
784 +&[is_zero.clone(), is_two.clone(), is_four.clone()],
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Description This 'multi_and' construction is redundant as it has only once argument.
Recommendation Consider removing it and using the argument directly.

Listing 11:

```
832 +multi_and(cs.namespace(|| "is_withdraw_nft_left"), &[
    ↪ is_withdraw_nft])?;
```

3.12 CVF-12

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation Unused code should be removed.

Listing 12:

```
834 +// // for fullexitNFT, we always select lhs
    +// let is_fullexit_nft = Expression::u64::<CS>(17 as u64);
    +// let is_fullexit_nft_left = multi_and(
    +//     cs.namespace(|| "is_fullexit_nft_left"),
    +//     &[is_transfer_tonew_nft, is_two.not()],
    +// )?;
840 +
```

3.13 CVF-13

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** circuit.rs

Recommendation A multi-or would be more readable.

Listing 13:

```
1208  Boolean::enforce_equal(
          cs.namespace(|| "0 <= fee_token < 2^(FEE_TOKEN_BIT_WIDTH-1)
              ↪ "),
1210 -    &op_data.fee_token.get_bits_le()[FEE_TOKEN_BIT_WIDTH - 1],
     +    &op_data.fee_token.get_bits_le()[FEE_TOKEN_BIT_WIDTH - 1],
         ↪ // b'0xxxxxx'
          &Boolean::constant(false),
      )?;
     +Boolean::enforce_equal(
     +    cs.namespace(|| "0 <= fee_token < 2^(FEE_TOKEN_BIT_WIDTH-2)
         ↪ "),
     +    &op_data.fee_token.get_bits_le()[FEE_TOKEN_BIT_WIDTH - 2],
         ↪ // b'00xxxxx'
     +    &Boolean::constant(false),
     +)?;
     +Boolean::enforce_equal(
1220 +    cs.namespace(|| "0 <= fee_token < 2^(FEE_TOKEN_BIT_WIDTH-3)
         ↪ "),
     +    &op_data.fee_token.get_bits_le()[FEE_TOKEN_BIT_WIDTH - 3],
         ↪ // b'000xxxx'
     +    &Boolean::constant(false),
     +)?;
```

3.14 CVF-14

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Recommendation Consider using a flag dedicated to regular accounts rather than check all possible other accounts.

Listing 14:

```
1633 +base_valid_flags.push(is_this_nft_account.not());
1950 +      base_valid_flags.push(is_this_nft_account.not());
2083 +is_valid_flags.push(is_this_nft_account.not());
2224 +is_valid_flags.push(is_this_nft_account.not());
2469 +lhs_valid_flags.push(is_this_nft_account.not());
2808 +lhs_valid_flags.push(is_this_nft_account.not());
2996 +rhs_valid_flags.push(is_this_nft_account.not());
```

3.15 CVF-15

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** circuit.rs

Recommendation This should be a named constant.

Listing 15:

```
1645 assert_eq!(pubdata_bits.len(), 46 * 8);
```

3.16 CVF-16

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** circuit.rs

Description The signature is verified in chunk 0, but "rhs" is verified in chunk 1, thus a malicious relayer may use correct recipient address in the chunk 0 to make the signature verification to succeed, but substitute another address in chunk 1, thus stealing the assets.
Recommendation Consider using 'op_data.eth_address' instead, and then, when verifying the 'chank1' enforce, that op_data.eth_address' is the same as 'cur.account.address'.

Listing 16:

```
2786 serialized_tx_bits.extend(rhs.account.address.get_bits_be());
```

3.17 CVF-17

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** circuit.rs

Description The rhs branch is not verified in chunk 0, where the signature is verified, thus it is possible to supply a malicious rhs.account.address in chunk1 where the signature is not verified, and thus add liquidity to a wrong pair.
Recommendation Consider taking all sensitive data from 'op.data' and verifying the latter against the signature.

Listing 17:

```
3277 serialized_tx_bits.extend(rhs.account.address.get_bits_be());

3368 chunk0_valid_flags.push(is_sig_verified.clone());
```

3.18 CVF-18

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** circuit.rs

Description The rhs branch is not verified in chunk 0, where the signature is verified, thus it is possible to supply a malicious rhs.account.address in chunk1 where the signature is not verified, and thus remove liquidity from a wrong pair.
Recommendation Consider taking all sensitive data from 'op.data' and verifying the latter against the signature.

Listing 18:

```
3849 serialized_tx_bits.extend(rhs.account.address.get_bits_be());

3923 chunk0_valid_flags.push(is_sig_verified.clone());
```

3.19 CVF-19

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Recommendation It is less error-prone to have a predicate that specifies which accounts are eligible for swaps, rather than which ones are NFT.

Listing 19:

```
4145 +is_this_nft_account: &Boolean,
```

3.20 CVF-20

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** circuit.rs

Description Is it really required to support this case?

Listing 20:

```
4633 +* 2/ account/pair_account is empty          // 2021.06.09
     ↪ pair_account can be non-empty in case account.eth_addr =
     ↪ cur.account.address
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Description This condition is redundant.

Listing 21:

```
4790 +is_account_empty.not(),
```

3.22 CVF-22

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Description The name is ambigiuos.
Recommendation Consider giving more descriptive names to flags.

Listing 22:

```
4795 +let is_either_valid = multi_or(
```

3.23 CVF-23

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Description Both arguments of the "multi_or" operation are conjunctions including "is_tx_valid" and "is_pair_account_empty".
Recommendation Consider moving them out of the disjunction, i.e. calcualte like: and (is_tx_valid, is_pair_account_empty, or (is_account_empty, is_address_same)).

Listing 23:

```
4797 +&[is_chunk0_valid.clone(), is_account_exist_and_valid.clone()],
```

3.24 CVF-24

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** circuit.rs

Description This operation doesn't charge fee, but at the same time it doesn't ensure that the fee amount is zero. Thus, a maliciour relayer may execute an NFT deposit with a non-zero fee amount, causing two critical problems: 1. The protocol state will become different from what one would expect looking at the pubdata 2. The relayer will mint tokens in L2 that are not backed by tokens in L1
Recommendation Consider adding a check similar the checks performed by full_exit and deposit operations.

Listing 24:

```
4864 +fn depositNFT<CS: ConstraintSystem<E>>(
```

3.25 CVF-25

- **Severity** Moderate
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Description There is no restriction on 'from_account_id' in this case.
Recommendation Probably it should be restricted to be a regular account.

Listing 25:

```
5078 +// chunk1: cur = from account
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Description We do not exactly modify the account, just check some equalities.

Listing 26:

```
5761 +// 9. in chunk2 we modify to account
```

3.27 CVF-27

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Recommendation This is confusing. It might be simpler to set 'approved_account_id' always to 0 except for exchanges.

Listing 27:

```
6065 +approved_account_id: op_data.to_account_id.clone(), // approved
     ↪ info should be reset after transfer
```

3.28 CVF-28

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation These variables can be computed in 'synthesize'.

Listing 28:

```
6962 +let is_chunk0 = Boolean::from(Expression::equals(

6968 +let is_chunk1 = Boolean::from(Expression::equals(

6974 +let is_chunk2 = Boolean::from(Expression::equals(
```

3.29 CVF-29

- **Severity** Major
- **Status** Info
- **Category** Unclear behavior
- **Source** circuit.rs

Description Default approved account ID value equals the value meaning 'everyone can initiate an exchange for this NFT'. It is not straightforward to observe why this does not lead to NFT duplicates or other attack vectors, as the countermeasure relies on special meaning for zero token ID and other fields.
Recommendation Consider using a different value for 'approved to everyone'.

Listing 29:

```
7340 +let anyone_can_exchange = CircuitElement::equals(
+     cs.namespace(|| "anyone can exchange"),
+     &op_data.nft_info.approved_account_id,
+     explicit_zero,
+)?;
```

3.30 CVF-30

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** circuit.rs

Description This literally makes impossible for an account with 0 id to own any NFT as it could be claimed by anyone.
Recommendation Consider forbidding the zero account id at all.

Listing 30:

```
7340 +let anyone_can_exchange = CircuitElement::equals(
+        cs.namespace(|| "anyone can exchange"),
+        &op_data.nft_info.approved_account_id,
+        explicit_zero,
+)?;
```

3.31 CVF-31

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** circuit.rs

Description Tree depths are global constants.
Recommendation Consider using them for readability.

Listing 31:

```
7690 +balance_tree_depth() + account_tree_depth() - 1,

7696 +nft_id_ce.get_bits_le()[balance_tree_depth()..].to_vec(),

7710 +nft_id_ce.get_bits_le()[..balance_tree_depth()].to_vec(),
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation This commented code should be removed.

Listing 32:

```
7783 +// let empty_root_padding =
+//     AllocatedNum::zero(cs.namespace(|| "allocate zero
   ↪ element for padding"))?;
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** operation.rs

Recommendation If the amount is assumed to be bounded in the protocol, it makes sense to document it here.

Listing 33:

```
27 +pub approved_amount: Option<E::Fr>,
```

```
65 +                E::Fr::from_str(&nft.approved_amount.to_u128().
   ↪ unwrap().to_string()).unwrap(),
```

3.34 CVF-34

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** operation.rs

Description This commented line is confusing.
Recommendation Consider removing it or explaining in more details what does it mean.

Listing 34:

```
84 +// pub nft_account_witness: NFTAccountWitness<E>,
```

3.35 CVF-35

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** operation.rs

Recommendation The comment should be modified.

Listing 35:

```
136 +pub fee2: Option<E::Fr>, // fee
```

3.36 CVF-36

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** nft_deposit.rs

Recommendation The approved token ID and approved amount should be set to 0, and approved account should be equal to 'from_account_id'.

Listing 36:

```
187 +.nft_info
```

3.37 CVF-37

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** nft_deposit.rs

Recommendation Default token should be a named constant.

Listing 37:

```
219 +let token = 0_u32;
```

3.38 CVF-38

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** nft_approve.rs

Description The operation name is very confusing, as it it not what is usually meant by "approve" regarding tokens. It looks more like placing a sell order.
Recommendation Consider choosing a better name.

Listing 38:

```
31 +struct NFTApproveData {
```

3.39 CVF-39

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** nft_approve.rs

Description Consider commenting if 'insert' handles duplicates

Listing 39:

```
197 +acc.nft_infos.insert(nft_bal_id, new_nft_info.clone());
```

3.40 CVF-40

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** nft_approve.rs

Description The declaration of the "nft_infos" field is out of scope for this audit, while it is clearly related to the NFT logic.

Listing 40:

```
197 +acc.nft_infos.insert(nft_bal_id, new_nft_info.clone());
```

3.41 CVF-41

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

Description These functions should emit some events.

Listing 41:

```
184 +function setMaxDepositAmount(uint128 _amount) external {

192 +function setWithdrawGasLimit(uint256 _gasLimit) external {

199 +function setWithdrawNFTGasLimit(uint256 _gasLimit) external {

204 +function setGenesisRootAndAddresses(bytes32 _genesisRoot,
    ↪ address _zkSyncCommitBlockAddress,
```

3.42 CVF-42

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSync.sol

Description Limiting individual deposits at small fraction of the maximum total token amount in the system could be very limiting for certain tokens whose total supply is spread among few holders.
Recommendation Consider limiting the total amount directly, by checking the contract's token balance.

Listing 42:

```
388 +require(deposit_amount <= maxDepositAmount, "fd011");

396 +require(deposit_amount <= maxDepositAmount, "fd013");
```

3.43 CVF-43

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

Description This requirement makes ‘_blockNumberFrom‘ redundant.

Listing 43:

```
559 +require(_blockNumberFrom == totalBlocksChecked + 1, "cw2");
```

3.44 CVF-44

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

Recommendation This constant should be named and documented.

Listing 44:

```
588 +gasReserveValue: 0xff
```

3.45 CVF-45

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

Recommendation This should have type IERC721.

Listing 45:

```
13 +address tokenContract;
```

3.46 CVF-46

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

Description Keys of these mappings are unclear.
Recommendation Consider documenting.

Listing 46:

```
30 +mapping(uint256 => L1Info) public infoMapL1;
   +mapping(address => mapping(uint256 => L2Info)) public infoMapL2
      ↪ ;

36 +mapping(uint256 => PendingWithdrawal) public pendingWithdrawals
      ↪ ;
```

3.47 CVF-47

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

Recommendation These low-level constant and functions should be moved to a utility library.

Listing 47:

```
43 +bytes constant ALPHABET = "123456789
      ↪ ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";

133 +function toBase58(bytes memory source) internal pure returns (
      ↪ string memory) {

158 +function toAlphabet(uint8[] memory indices) internal pure
      ↪ returns (string memory) {

166 +function reverse(uint8[] memory input) internal pure returns (
      ↪ uint8[] memory) {
```

3.48 CVF-48

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

Recommendation These should be named constant(s).

Listing 48:

```
50 +constructor() public ERC721("ZKBox", "ZKBox") {

56 +     _name = "ZKBox";
   +     _symbol = "ZKBox";
```

3.49 CVF-49

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

Description Referring to "zkSync" in identifiers is weird.
Recommendation Consider replacing with "ZKSwap" or making identifiers neutral such as "setCoreAddress".

Listing 49:

```
62 +function setZkSyncAddress(address _zksyncAddress) external {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** ZKBoxNFT.sol

Description There is no explicit check that "_zksyncAddress" is not zero, thus technically it is possible to initialize several times.
Recommendation Consider adding such check.

Listing 50:

```
63 +require(zksCore == address(0), "ZKBoxNFT: already initialized")
     ↪ ;
   +zksCore = _zksyncAddress;
```

3.51 CVF-51

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

Recommendation ‘c‘ is a very ambiguous name.

Listing 51:

```
73 +function onDeposit(IERC721 c, uint256 tokenId, address addr)
   ↪ external onlyZksCore returns (Operations.DepositNFT memory
   ↪ ) {
```

3.52 CVF-52

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

Description This variable is redundant as it holds the same value as "c".

Listing 52:

```
79 +address tokenContractAddress = address(c);
```

3.53 CVF-53

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

Description There is already a variable for "address(c)".

Listing 53:

```
89 +tokenContract: address(c),
```

3.54 CVF-54

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** ZKBoxNFT.sol

Description Inputs of this function are known to always be 34 bytes long. This allows significantly optimizing the function by treating the input as an 272-bit integer and converting it into base58 representation using division and modulo operations. The first few iterations will require dealing with numbers longer than 256 bits, but such operations are still quite cheap in ethereum. The vast majority of iterations would deal with numbers that already fit into 256 bits.

Listing 54:

```
133 +function toBase58(bytes memory source) internal pure returns (
    ↪ string memory) {
```

3.55 CVF-55

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

Description This function copies bytes one by one, which is inefficient.
Recommendation Consider optimizing using the following function that reverses bytes in a 32-bytes word: function reverseBytes (uint x) public pure returns (uint) { x = x « 128 | x » 128; x = (x & 0xFFFFFFFFFFFFFFFF0000000000000000FFFFFFFFFFFFFFFF) « 64 | x » 64 & 0xFFFFFFFFFFFFFFFF0000000000000000FFFFFFFFFFFFFFFF; x = (x & 0xFFFFFFFF00000000FFFFFFFF00000000FFFFFFFF00000000FFFFFFFF) « 32 | x » 32 & 0xFFFFFFFF00000000FFFFFFFF00000000FFFFFFFF00000000FFFFFFFF; x = (x & 0xFFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF) « 16 | x » 16 & 0xFFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF; return (x & 0xFF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF) « 8 | x » 8 & 0xFF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF00FF; }"

Listing 55:

```
166 +function reverse(uint8[] memory input) internal pure returns (
    ↪ uint8[] memory) {
```

3.56 CVF-56

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

Description This function should emit some event.

Listing 56:

```
175 +function setContractURI(string memory contractURI_) public
    ↪ onlyOwner {
```

3.57 CVF-57

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

Description Conversion to "bytes28" perform another shift here. Using "uint224" instead of "bytes28" for withdrawal keys would avoid extra shifts.

Listing 57:

```
186 +return bytes28((uint224(addr) | (uint224(globalId) << 160)));
```

3.58 CVF-58

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

Recommendation ‘packWithdrawKey‘ should be used here.

Listing 58:

```
191 +bytes28 withdrawKey = bytes28((uint224(addr) | (uint224(
    ↪ globalId) << 160)));
```

3.59 CVF-59

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

Description The value of the "numOfPendingWithdrawals" variable is read from the storage twice.
Recommendation Consider using the "++" operator to read it once.

Listing 59:

```
227 +pendingWithdrawals[numOfPendingWithdrawals] = PendingWithdrawal
    ↪ (wd.target, wd.globalId);
+numOfPendingWithdrawals = numOfPendingWithdrawals + 1;
```

3.60 CVF-60

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

Recommendation ‘startIndex + toProcess‘ can be replaced with ‘firstPendingWithdrawal‘.

Listing 60:

```
244 +for (uint32 i = startIndex; i < startIndex + toProcess; ++i) {
```

3.61 CVF-61

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

Recommendation The first parameter should have type IERC721.

Listing 61:

```
259 +function onWithdraw(address target, uint64 globalId) external
    ↪ onlyZksCore returns (address, uint256) {
```

3.62 CVF-62

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

Description Why one of these functions public and the other external?
Recommendation Consider using consistent modifiers.

Listing 62:

```
268 +function tokenURI(uint256 tokenId) public view returns (string
    ↪ memory) {

275 +function getContentHash(uint256 _tokenId) external view returns
    ↪  (bytes32) {
```

3.63 CVF-63

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** nft_mint.rs

Description Overflow is possible here.
Recommendation Consider asserting the ID fits 32 bits explicitly

Listing 63:

```
56 +seq_id: seq_id.to_u32().unwrap(),
```

3.64 CVF-64

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** nft_mint.rs

Recommendation This should be a named constant.

Listing 64:

```
61 +approved_token_id: 0,
```

3.65 CVF-65

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** nft_mint.rs

Description Overflow is possible here.
Recommendation Consider asserting the fee fits 128 bits explicitly.

Listing 65:

```
67 +fee: op.tx.fee.to_u128().unwrap(),
```

3.66 CVF-66

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** nft_mint.rs

Description Variable-length arrays are error prone as they should be encoded properly.
Recommendation Consider using explicit bound for the URI length.

Listing 66:

```
97 +pubdata_bits.extend(uri);
```

3.67 CVF-67

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** nft_mint.rs

Recommendation It should be asserted that overflow does not happen at casting.

Listing 67:

```
185 +let (account_path, balance_path) = get_audits(tree, nft_acc_id,
    ↪  nft_bal_id as u32);
```

3.68 CVF-68

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** allocated_structures.rs

Recommendation Unused code should be removed.

Listing 68:

```
90 +// if hash_ce.get_number().get_value().is_some() {
   +//     println!(
   +//         "[circuit] hash: {:?}",
   +//         hash_ce.get_number().get_value().unwrap()
   +//     );
   +// }
```

3.69 CVF-69

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** allocated_structures.rs

Description This renaming seems incomplete.
Recommendation Consider using consistent naming for all related variables.

Listing 69:

```
208 -pub balance2_audit_path: Vec<AllocatedNum<E>>,

214 +pub subtree_audit_path2: Vec<AllocatedNum<E>>,

275     let balance2_audit_path = utils::allocate_numbers_vec(

277         &operation_branch.witness.balance2_subtree_path,

300 +       subtree_audit_path2: balance2_audit_path,
```

3.70 CVF-70

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** allocated_structures.rs

Description Aren't the pair balances in reserve tokens equal to the reserve values? If yes one of them is redundant.

Listing 70:

```
353 +pub r0: CircuitElement<E>,          // pair.reserve0
    +pub r1: CircuitElement<E>,          // pair.reserve1
    +pub b0: CircuitElement<E>,          // pair.balance[token0]
    +pub b1: CircuitElement<E>,          // pair.balance[token1]
```

3.71 CVF-71

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** Storage.sol

Description As NFT global IDs are assigned sequentially.
Recommendation It would be more efficient to use a bit mask here, i.e. store flags for 256 tokens in a single slot.

Listing 71:

```
100 +mapping(uint64 => bool) public nft_exited;
```

3.72 CVF-72

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

Description These values are actually only the default values, while actual values could be set by the governance.
Recommendation Consider adding "DEFAULT_" prefix to the constant names.

Listing 72:

```
 11 +uint256 constant ERC20_WITHDRAWAL_GAS_LIMIT = 350000;

 14 +uint256 constant ERC721_WITHDRAWAL_GAS_LIMIT = 350000;
```

3.73 CVF-73

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Config.sol

Description It is a bit unusual that the maximum ID value is a power of two, rather than a power of two minus one. Are NFT IDs one-based? If so, consider defining the "MIN_NFT_ID" constant as well and using it when checking for validity of an NFT ID.

Listing 73:

```
53 +uint64 constant MAX_NFT_ID = 2**(27+16);
```

3.74 CVF-74

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Config.sol

Description These fields occupy 39 bytes rather than 71.

Listing 74:

```
90 +/// (uint8 isNFTWithdraw uint8 addToPendingWithdrawalsQueue,
   ↪ uint64 globalId, uint32 creator,
+//  uint32 seqId, address _toAddr, uint8 isValid)
+uint256 constant ONCHAIN_WITHDRAWAL_NFT_BYTES = 71;
```

3.75 CVF-75

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Config.sol

Description It is a bit unusual that the maximum deposit amount is a power of two, rather than a power of two minus one. Is there any reasoning for this?

Listing 75:

```
121 +uint128 constant DEFAULT_MAX_DEPOSIT_AMOUNT = 2 ** 85;
```

3.76 CVF-76

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

Description This field doesn't seem to be ever read.
Recommendation Consider removing it.

Listing 76:

```
220 +bool valid;  //confirm the necessity of this field
```

3.77 CVF-77

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Description Solidity compiler is smart enough to calculate constant expressions at compile time, even those expressions that calculate hash functions.
Recommendation Consider initializing constants using expressions rather than hardcoded values.

Listing 77:

```
53 +bytes4 private constant _ERC721_RECEIVED = 0x150b7a02;

69 +bytes4 private constant _INTERFACE_ID_ERC721 = 0x80ac58cd;

78 +bytes4 private constant _INTERFACE_ID_ERC721_METADATA = 0
    ↪ x5b5e139f;

83 +bytes4 private constant _INTERFACE_ID_ERC165 = 0x01ffc9a7;

93 +bytes4 private constant _INTERFACE_ID_ERC721_ENUMERABLE = 0
    ↪ x780e9d63;
```

3.78 CVF-78

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Recommendation This value could be obtained as "IERC721Receiver(to).onERC721Received.selector". No need for a separate constant.

Listing 78:

```
53 +bytes4 private constant _ERC721_RECEIVED = 0x150b7a02;
```

3.79 CVF-79

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Description These functions wouldn't be needed if the corresponding variable would be declared as public and named appropriately.

Listing 79:

```
135 +function name() public view  returns (string memory) {
142 +function symbol() public view  returns (string memory) {
```

3.80 CVF-80

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Description This event is logged even if the status has not changed.

Listing 80:

```
244 +emit ApprovalForAll(msg.sender, operator, approved);
```

3.81 CVF-81

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** ERC721.sol

Description Returned values are ignored.

Listing 81:

```
391 +_holderTokens[to].add(tokenId);
393 +_tokenOwners.set(tokenId, to);
418 +_holderTokens[owner].remove(tokenId);
    +_tokenOwners.remove(tokenId);
446 +_holderTokens[from].remove(tokenId);
    +_holderTokens[to].add(tokenId);
449 +_tokenOwners.set(tokenId, to);
```

3.82 CVF-82

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Description This argument is redundant as it could be derived from the "tokenId" value.
Recommendation Consider removing this argument.

Listing 82:

```
436 +address from,
```

3.83 CVF-83

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ERC721.sol

Description This call emits an Approval event, which is weird.

Listing 83:

```
444 +_approve(address(0), tokenId);
```

3.84 CVF-84

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ERC721.sol

Recommendation These functions should emit some events.

Listing 84:

```
463 +function _setTokenURI(uint256 tokenId, string memory _tokenURI)
    ↪ internal {

474 +function _setBaseURI(string memory baseURI_) internal {
```

3.85 CVF-85

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC721.sol

Recommendation This call could be performed in a normal way like this: return IERC721Receiver (to).onERC721Received(msg.sender, from, tokenId, _data) == _ERC721_RECEIVED;"

Listing 85:

```
498 +(bool success, bytes memory returndata) = to.call(abi.
    ↪ encodeWithSelector(
+           IERC721Receiver(to).onERC721Received.selector,
500 +           msg.sender,
+           from,
+           tokenId,
+           _data
+       ));
```

3.86 CVF-86

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

Recommendation This parameter should be indexed. As there couldn't be more than three indexed parameters consider either making some other parameter not indexed or indexing a hash of token+tokenId, as tokenId doesn't make much sense without token contract address.

Listing 86:

```
40 +uint256   tokenId,
```

3.87 CVF-87

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

Recommendation Consider explaining why this overflow is impossible in practice, or even better, add an explicit overflow check.

Listing 87:

```
97 +// NOTE: theoretically possible overflow of (_start + 0x8)

210 +// NOTE: theoretically possible overflow of (_offset + 8)
```

3.88 CVF-88

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

Description Despite the name, this function doesn't convert the whole "_bytes" into uint64, but extracts a uint64 value from it.
Recommendation Consider renaming.

Listing 88:

```
98 +function bytesToUInt64(bytes memory _bytes, uint256 _start)
     ↪ internal pure returns (uint64 r) {
```

3.89 CVF-89

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

Description The first parameter is redundant as it can be derived from the previous event.

Listing 89:

```
15 +event OwnershipTransferred(address indexed previousOwner,
     ↪ address indexed newOwner);
```

3.90 CVF-90

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

Recommendation Events are usually named via nouns, such as "OwnershipTransfer" or just "Owner".

Listing 90:

```
15 +event OwnershipTransferred(address indexed previousOwner,
     ↪ address indexed newOwner);
```

3.91 CVF-91

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

Description This variable is redundant, as "msg.sender" is cheaper to access than a local variable.

Listing 91:

```
21 +address msgSender = msg.sender;
```

3.92 CVF-92

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

Description This check is redundant as it is always possible to transfer ownership to a dead address.
Recommendation Consider dropping it as well as the ‘renounceOwnership‘ function.

Listing 92:

```
81 +require(newOwner != address(0), "Ownable: new owner is the zero
   ↪ address");
```

3.93 CVF-93

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

Description The role of this interface and the semantics of its functions is unclear.
Recommendation Consider documenting.

Listing 93:

```
7 +interface IZKBoxNFT {
```

3.94 CVF-94

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

Description The semantics of the returned values is unclear.
Recommendation Consider documenting.

Listing 94:

```
12 +function onWithdraw(address target, uint64 globalId) external
    ↪ returns (address, uint256);
```

3.95 CVF-95

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

Recommendation The type of this field should be more specific.

Listing 95:

```
17 +address tokenContract;
```

3.96 CVF-96

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** nft_withdraw.rs

Description Variable-length URI make collisions possible.
Recommendation Use fixed-length URI.

Listing 96:

```
105 +// append_be_fixed_width(
    +//     &mut pubdata_bits,
    +//     &self.args.nft_info.uri.unwrap(),
    +//     URI_BIT_WIDTH,
    +// );
110 +
    +let uri = self
    +    .args
    +    .nft_info
    +    .uri
    +    .iter()
    +    .map(|b| b.unwrap())
    +    .collect::<Vec<bool>>();
    +pubdata_bits.extend(uri);
```

3.97 CVF-97

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** nft_withdraw.rs

Description These constants should be named.

Listing 97:

```
224 +    owner: 0,
    +    approved_account_id: 0,
    +    approved_token_id: 0,
    +    approved_amount: 0u64.into(),
    +};
```

3.98 CVF-98

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** nft_withdraw.rs

Recommendation This comment should be removed to avoid confusion.

Listing 98:

```
241 +// bal.value = Fr::zero();
```

3.99 CVF-99

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** nft_transfer.rs

Recommendation It should be asserted there is no overflow.

Listing 99:

```
69 +let fee = op.tx.fee.to_u128().unwrap();
```

3.100 CVF-100

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** nft_transfer.rs

Recommendation This should be a named constant.

Listing 100:

```
193 +new_nft_info.approved_amount = BigUint::from(0_u8);
```

3.101 CVF-101

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** nft_transfer_to_new.rs

Recommendation Consider asserting that the value fits 128 bits for readability.

Listing 101:

```
71 +let fee = op.tx.fee.to_u128().unwrap();
```

3.102 CVF-102

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** nft_transfer_to_new.rs

Recommendation Default constants should be named.

Listing 102:

```
220 +new_nft_info.approved_token_id = 0;
    +new_nft_info.approved_amount = BigUint::from(0_u8);
```

3.103 CVF-103

- **Severity** Major
- **Status** Fixed
- **Category** Procedural
- **Source** nft_full_exit.rs

Description Uri has fixed bit width in the protocol description.
Recommendation Consider using fixed length here or at least prefix the uri with the length.

Listing 103:

```
111 +let uri = self
    +     .args
    +     .nft_info
    +     .uri
    +     .iter()
    +     .map(|b| b.unwrap())
    +     .collect::<Vec<bool>>();
    +pubdata_bits.extend(uri);
```

3.104 CVF-104

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** nft_full_exit.rs

Recommendation These constants should be named.

Listing 104:

```
241 +     owner: 0,
    +     approved_account_id: 0,
    +     approved_token_id: 0,
    +     approved_amount: 0u64.into(),

258 +let (acc_wit, _, bal_val) = get_pair_account_witness(tree, data
        ↪ .from_account_id, 0u32);

299 +let token = 0_u32;
```

3.105 CVF-105

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** nft_full_exit.rs

Description This case is not documented in the protocol doc.

Listing 105:

```
249 +// in case this nft does not exist (bal = 0), prover should not
        ↪  update branch
250 +let (_, bal_val) = get_account_witness(tree, nft_acc_id,
        ↪ nft_bal_id as u32);
    +if bal_val == Fr::zero() {
    +     success = false;
    +     args.b = Some(Fr::zero())
    +} else {
    +     args.b = Some(Fr::one())
    +}
    +
```

3.106 CVF-106

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** nft_full_exit.rs

Recommendation Consider commenting what ‘a‘ means here.

Listing 106:

```
335 +args.a = if from.witness.account_witness.address.is_some()
    +     && from.witness.account_witness.address.unwrap() == data.
       ↪ from_addr
    +{
    +     Some(Fr::one())
    +} else {
340 +     Some(Fr::zero())
    +};
```

3.107 CVF-107

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** nft_full_exit.rs

Description This comment is probably incorrect as args.c is not used.

Listing 107:

```
343 +// args.c indicates whether we should update nft branch
    +// args.c should equal to plasma update status
```

3.108 CVF-108

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** nft_exchange.rs

Recommendation Default token ID should be a named constant.

Listing 108:

```
214 +new_nft_info.approved_token_id = 0;
```

3.109 CVF-109

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

Description This function is quite inefficient.
Recommendation Consider using an approach suggested here: https://stackoverflow.com/a/71095692/2038768

Listing 109:

```
12 +function toString(uint256 value) internal pure returns (string
    ↪ memory) {
```

3.110 CVF-110

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

Description This loop is redundant.
Recommendation Just allocate a string of 77 chars, fill it right to left and then shop the unused prefix in-place.

Listing 110:

```
21 +while (temp != 0) {
+     digits++;
+     temp /= 10;
+}
```

3.111 CVF-111

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

Description This should be executed only when toDeleteIndex != lastIndex.

Listing 111:

```
87 +bytes32 lastvalue = set._values[lastIndex];

90 +set._values[toDeleteIndex] = lastvalue;

92 +set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are
    ↪ 1-based
```

3.112 CVF-112

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

Recommendation This can be just ‘valueIndex‘.

Listing 112:

```
92 +set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are
   ↪ 1-based
```

3.113 CVF-113

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

Description The term "size" is more common for sets than "length".

Listing 113:

```
116 +function _length(Set storage set) private view returns (uint256
   ↪ ) {
```

3.114 CVF-114

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

Description This check is redundant, as Solidity compiler will do the same check at the next line.

Listing 114:

```
131 +require(set._values.length > index, "EnumerableSet: index out
   ↪ of bounds");
```

3.115 CVF-115

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

Description This structure and related set of functions is redundant, as "Set" is already a "bytes32" set.
Recommendation Consider removing this structure and corresponding functions, renaming "Set" to "Bytes32Set" and making "Set" functions internal rather than private.

Listing 115:

```
137 +struct Bytes32Set {
    +     Set _inner;
    +}
```

3.116 CVF-116

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

Recommendation It would be cheaper to use EXTCODEHASH instead of EXTCODESIZE. Note, that EXTCODEHASH returns different values for a non-existing account and existing non-contract account, so the returned value should be checked against both values. See EIP-1052 for details: https://eips.ethereum.org/EIPS/eip-1052

Listing 116:

```
33 +size := extcodesize(account)
```

3.117 CVF-117

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** ZkSyncCommitBlock.sol

Description Note that this way of hashing withdrawals may give the same results for withdrawals of different type, in the case the other data is identical.
Recommendation Consider prefixing the operation type to the input.

Listing 117:

```
465 +withdrawalsDataHash = keccak256(abi.encode(withdrawalsDataHash,
    ↪  addToPendingWithdrawalsQueue, fullExitNFTData.owner,
    ↪ fullExitNFTData.globalId));
```

3.118 CVF-118

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** utils.rs

Recommendation Consider documenting these functions and their usage

Listing 118:

```
199 +pub fn generate_dummy_balance_branch(
```

```
213 +pub fn generate_dummy_operation_branch(
```

3.119 CVF-119

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** utils.rs

Description This name is confusing and harms readability.
Recommendation Consider calling it ‘get_account_and_balance_witness‘.

Listing 119:

```
479 pub fn get_pair_account_witness(
```

3.120 CVF-120

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

Recommendation The mask size should be a constant defined in a Plonk config file.

Listing 120:

```
48 +uint256 mask = (~uint256(0)) >> 3;
```

3.121 CVF-121

- **Severity** Major
- **Status** Info
- **Category** Flaw
- **Source** nft_exit_circuit.rs

Description This value is field-dependent.
Recommendation Consider making it a method for Fr.

Listing 121:

```
223 +hash_result[0] &= 0x1f; // temporary solution, this nullifies
    ↪ top bits to be encoded into field element correctly
```