ABDK CONSULTING

SMART CONTRACT AUDIT

Spectral

Spectral.finance

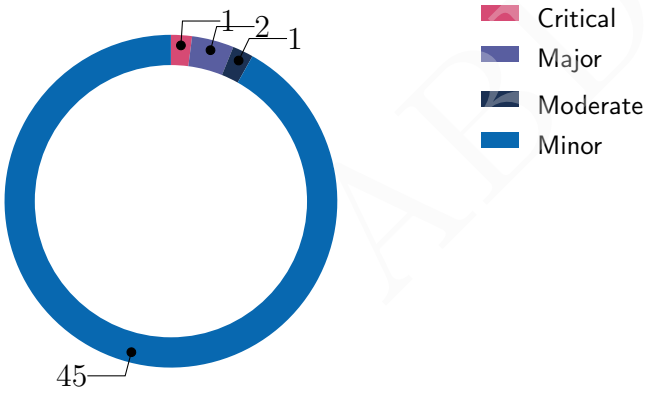
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
20th May 2022

We've been asked to review 6 files in a Github repository. We found 1 critical, 2 major, and a few less important issues. All critical and major issues have been fixed.



Findings

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

ID	Severity	Category	Status
CVF-28	Minor	Documentation	Info
CVF-29	Minor	Bad datatype	Info
CVF-30	Minor	Bad datatype	Info
CVF-31	Minor	Procedural	Info
CVF-32	Minor	Suboptimal	Fixed
CVF-33	Minor	Documentation	Fixed
CVF-34	Minor	Documentation	Info
CVF-35	Minor	Suboptimal	Fixed
CVF-36	Minor	Suboptimal	Info
CVF-37	Minor	Procedural	Fixed
CVF-38	Minor	Procedural	Fixed
CVF-39	Major	Suboptimal	Fixed
CVF-40	Minor	Suboptimal	Info
CVF-41	Minor	Bad naming	Info
CVF-42	Minor	Bad datatype	Info
CVF-43	Minor	Bad datatype	Info
CVF-44	Minor	Documentation	Fixed
CVF-45	Minor	Bad datatype	Info
CVF-46	Minor	Procedural	Info
CVF-47	Minor	Bad naming	Info
CVF-48	Minor	Suboptimal	Fixed
CVF-49	Minor	Procedural	Fixed

Contents

1 Document properties **7**

2 Introduction **8**
2.1 About ABDK . . . 8
2.2 Disclaimer . . . 8
2.3 Methodology . . . 8

3 Detailed Results **10**
3.1 CVF-1 . . . 10
3.2 CVF-2 . . . 10
3.3 CVF-3 . . . 10
3.4 CVF-4 . . . 11
3.5 CVF-5 . . . 11
3.6 CVF-6 . . . 11
3.7 CVF-7 . . . 12
3.8 CVF-8 . . . 12
3.9 CVF-9 . . . 13
3.10 CVF-10 . . . 13
3.11 CVF-11 . . . 14
3.12 CVF-12 . . . 14
3.13 CVF-13 . . . 14
3.14 CVF-14 . . . 15
3.15 CVF-15 . . . 15
3.16 CVF-16 . . . 15
3.17 CVF-17 . . . 16
3.18 CVF-18 . . . 17
3.19 CVF-19 . . . 17
3.20 CVF-20 . . . 18
3.21 CVF-21 . . . 18
3.22 CVF-22 . . . 18
3.23 CVF-23 . . . 19
3.24 CVF-24 . . . 19
3.25 CVF-25 . . . 20
3.26 CVF-26 . . . 20
3.27 CVF-27 . . . 20
3.28 CVF-28 . . . 21
3.29 CVF-29 . . . 21
3.30 CVF-30 . . . 21
3.31 CVF-31 . . . 22
3.32 CVF-32 . . . 22
3.33 CVF-33 . . . 22
3.34 CVF-34 . . . 23
3.35 CVF-35 . . . 23
3.36 CVF-36 . . . 23
3.37 CVF-37 . . . 24

3.38 CVF-38 24
3.39 CVF-39 24
3.40 CVF-40 25
3.41 CVF-41 25
3.42 CVF-42 26
3.43 CVF-43 26
3.44 CVF-44 26
3.45 CVF-45 27
3.46 CVF-46 27
3.47 CVF-47 27
3.48 CVF-48 28
3.49 CVF-49 28

1 Document properties

Version

Version	Date	Author	Description
0.1	May 19, 2022	D. Khovratovich	Initial Draft
0.2	May 19, 2022	D. Khovratovich	Minor revision
1.0	May 20, 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/INFC.sol
- interfaces/IScoracle.sol
- interfaces/ITicksets.sol
- NFC.sol
- Scoracle.sol
- Ticksets.sol

The fixes were provided in the e110d2f 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** Procedural
- **Source** Scoracle.sol

Recommendation Should be "^0.8.0" unless there is something special about this particular version. Also relevant for the next files:Ticksets.sol, ITicksets.sol, NFC.sol, IScoracle.sol, ISpectralComptrollerAdmin.sol.
Client Comment I prefer explicitly stating solidity versions for two reasons: 1. Consistency between team members 2. Easier to verify contracts on etherscan

Listing 1:

```
2 pragma solidity 0.8.11;
```

3.2 CVF-2

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

Description We didn't review these files.
Client Comment Acknowledged.

Listing 2:

```
6 import {Chainlink} from "@chainlink/contracts/src/v0.8/Chainlink
    ↪ .sol";
import {ChainlinkClient} from "@chainlink/contracts/src/v0.8/
    ↪ ChainlinkClient.sol";
```

3.3 CVF-3

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

Recommendation The types of these variables could be more specific.
Client Comment Non-issue.

Listing 3:

```
22 address public chainlinkNode; //chainlink node endpoint
   address public oracle; //address of chainlink oracle contract
```

3.4 CVF-4

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

Description The semantics of keys and values in these mappings is unclear.
Recommendation Consider documenting.
Client Comment Fixed in commit ac8b276.

Listing 4:

```
34 mapping(address => mapping(bytes32 => ScoreData)) public
   ↪ scoreData;
mapping(bytes32 => Request) public reqIdAddress;
mapping(bytes32 => string) public scoreTypesJobIds;
mapping(address => uint256) internal nonces;
```

3.5 CVF-5

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

Recommendation The types of these arguments could be more specific.
Client Comment Non-issue.

Listing 5:

```
40 address _chainlinkNode,
address _chainlinkOracle,
address _link,
```

3.6 CVF-6

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Scoracle.sol

Recommendation These values should be named constants.
Client Comment Fixed in commit d45162b.

Listing 6:

```
46 minScore = 350;
maxScore = 850;
```

3.7 CVF-7

- **Severity** Moderate
- **Status** Fixed
- **Category** Suboptimal
- **Source** Scoracle.sol

Description The value of this argument is not included into a message whose signature is verified. Thus, a caller of this function could set arbitrary value for this argument and the signature would still be valid. Moreover, a malicious user may front run transaction, that calls this function, and invoke the same function with the same signature, but with different value of this argument, thus wasting the nonce.
Recommendation Consider either including the value of this argument into a message being signed, or using come other approach to prevent tampering with this argument.
Client Comment Fixed in commit 57c42b8.

Listing 7:

```
69 bytes32 _scoreTypeJobId,
```

3.8 CVF-8

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

Recommendation It is a good practice to make payable functions external, rather than public, to avoid double counting of incoming ether. Lets consider situation when a public payable function A is called with 1 ETH and this function calls a public payable function B of the same contract. Both functions would see msg.value == 1 ether, and each function could thing that this 1 ether was passed to it, while actually it was passed only to the function A.
Client Comment Fixed in commit 545c353.

Listing 8:

```
71 ) public payable virtual override {
```

3.9 CVF-9

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

Description These events are emitted even if nothing actually changed.
Client Comment Acknowledged.

Listing 9:

```
133 emit ScoreTypeAdded(_scoreTypeJobId, _scoreTypeName);

143 emit ScoreTypeDeactivated(_scoreTypeJobId);

152 emit ChainlinkNodeUpdated(chainlinkNode);

161 emit ChainlinkOracleUpdated(oracle);

170 emit BaseFeeUpdated(baseFee);

179 emit ChainlinkFeeUpdated(chainlinkFee);
```

3.10 CVF-10

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

Description This check is redundant, as it will anyway be performed by the "transferFrom" call below.
Client Comment Fixed in commit 6681a6a.

Listing 10:

```
188 require(token.allowance(msg.sender, address(this)) >= _amount, "
    ↪ Allowance lower than needed");
```

3.11 CVF-11

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

Recommendation This could be simplified as: exists = bytes(scoreTypesJobIds[_scoreTypeJobId]).length > 0;
Client Comment Fixed in commit 0daa3b6.

Listing 11:

```
241 if (bytes(scoreTypesJobIds[_scoreTypeJobId]).length > 0) {
        exists = true;
    } else {
        exists = false;
    }
```

3.12 CVF-12

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

Recommendation Usually constants are named in UPPERCASE.
Client Comment Fixed in commit 6d76f51.

Listing 12:

```
18 uint256 private constant minTickSize = 50;
   uint256 private constant maxTickSize = 300;
```

3.13 CVF-13

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Ticksets.sol

Recommendation These values should be named constants.
Client Comment Fixed in commit 6d76f51.

Listing 13:

```
18 uint256 private constant minTickSize = 50;
   uint256 private constant maxTickSize = 300;
```

3.14 CVF-14

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Ticksets.sol

Recommendation The type of this variable should be "IScoracle".
Client Comment Fixed in commit 445663e.

Listing 14:

```
23 address public scoracleAddress;
```

3.15 CVF-15

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Ticksets.sol

Recommendation The argument type should be "IScorable".
Client Comment Fixed in commit 6d76f51.

Listing 15:

```
25 constructor(address _scoracleAddress) {

73 function updateScoracleAddress(address _scoracleAddress)
     ↪ external override onlyOwner {
```

3.16 CVF-16

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

Recommendation This check should be performed earlier.
Client Comment Fixed in commit e835ae2.

Listing 16:

```
43 require(_tickEnds.length > 1, _errorMsg("
     ↪ TICK_ENDS_LENGTH_LOWER_THAN_2", msg.sender));
```

3.17 CVF-17

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

Recommendation This could code be simplified like this:
uint256 tickBegin = mnScore; for (uint256 i = 0; i < _tickEnds.length; i++) { uint256 tickEnd = _tickEnds [i]; uint256 tickSize = tickEnd - tickBegin; require (tickSize >= minTickSize && tickSize <= maxTickSize, "TICK_SIZE_FAIL"); tickBegin = tickEnd; }
Client Comment Fixed in commit e835ae2.

Listing 17:

```
47  require(
        _tickEnds[0] > minScore && _tickEnds[_tickEnds.length - 1]
            ↪ <= maxScore,
        _errorMsg("WRONG_SCORE_BOUNDS", msg.sender)
50  );

52  for (uint256 i = 0; i < _tickEnds.length - 1; i++) {
        if (i == 0 && (_tickEnds[i] - minScore < minTickSize ||
            ↪ _tickEnds[i] - minScore > maxTickSize)) {
            revert("TICK_SIZE_FAIL");
        } else if (
            i != 0 &&
            (_tickEnds[i + 1] - _tickEnds[i] < minTickSize ||
                ↪ _tickEnds[i + 1] - _tickEnds[i] > maxTickSize)
        ) {
            revert("TICK_SIZE_FAIL");
60      }
    }
```

3.18 CVF-18

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** Ticksets.sol

Description For i == 0 this checks the interval between minScore and _tickEnds [0]. For i == 1 this checks the interval between _tickEnds [1] and _tickEnds [2]. Thus, teh interval between _tickEnds [0] and _tickEnds [1] is never checked.
Client Comment Fixed in commit e835ae2.

Listing 18:

```
53 if (i == 0 && (_tickEnds[i] - minScore < minTickSize ||
   ↪ _tickEnds[i] - minScore > maxTickSize)) {
     revert("TICK_SIZE_FAIL");
   } else if (
     i != 0 &&
     (_tickEnds[i + 1] - _tickEnds[i] < minTickSize || _tickEnds[
       ↪ i + 1] - _tickEnds[i] > maxTickSize)
   ) {
     revert("TICK_SIZE_FAIL");
60 }
```

3.19 CVF-19

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

Description String error messages are suboptimal.
Recommendation Consider using named errors.
Client Comment Fixed in commit d978c2d.

Listing 19:

```
54 revert("TICK_SIZE_FAIL");

59 revert("TICK_SIZE_FAIL");
```

3.20 CVF-20

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** Ticksets.sol

Description Here, the whole "_tickSet_tickEnds[tempID]" array is read from the storage, while it was just written there one line above.
Recommendation Consider using the "_tickEnds" array instead.
Client Comment Fixed in commit cc0c515.

Listing 20:

```
63 emit TickSetCreated(tempID, _tickSet_tickEnds[tempID]);
```

3.21 CVF-21

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

Description This event is emitted even if nothing actually changed.
Client Comment Acknowledged.

Listing 21:

```
75 emit ScoracleUpdated(_scoracleAddress);
```

3.22 CVF-22

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

Description This function uses linear search which is suboptimal.
Recommendation Consider using binary search instead.
Client Comment Acknowledged.

Listing 22:

```
86 function getTickUsingTickSet(
```

3.23 CVF-23

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

Description Here the whole array is read into memory, while only part of the array will actual be needed.
Recommendation Consider reading only the needed elements.
Client Comment Acknowledged.

Listing 23:

```
100 uint256[] memory tickEnds = _tickSet_tickEnds[_tickSetId];
```

3.24 CVF-24

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

Recommendation This look could be optimized as: if (score >= minScore) { for (uint256 i = 0; i < tickEnds.length; i++) { if (score <= tickEnds[i]) { tick = i + 1; break; } } }
Client Comment Fixed in commit 3427cc1.

Listing 24:

```
102 for (uint256 i = 0; i < tickEnds.length; i++) {
        if (i == 0 && score >= minScore && score <= tickEnds[i]) {
            tick = i + 1;
            break;
        } else if (i != 0 && score > tickEnds[i - 1] && score <=
            ↪ tickEnds[i]) {
            tick = i + 1;
            break;
        }
110 }
```

3.25 CVF-25

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

Recommendation Events are usually named via nouns, such as "TicketSet" or "Scoracle".
Client Comment Acknowledged.

Listing 25:

```
12 event TickSetCreated(uint256 indexed tickSetId, uint256[]
   ↪ tickEnds);

18 event ScoracleUpdated(address scoracleAddress);
```

3.26 CVF-26

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

Recommendation The parameter should be indexed.
Client Comment Fixed in commit 30a5c4f.

Listing 26:

```
18 event ScoracleUpdated(address scoracleAddress);
```

3.27 CVF-27

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

Recommendation The parameter type should be "IScoracle".
Client Comment Acknowledged.

Listing 27:

```
18 event ScoracleUpdated(address scoracleAddress);
```

3.28 CVF-28

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

Description The semantics of the returned value is unclear.
Recommendation Consider giving a descriptive name to the returned value and/or adding a documentation comment.
Client Comment Acknowledged.

Listing 28:

```
22 function createTickSet(uint256[] memory _tickEnds) external
   ↪ returns (bool);
```

3.29 CVF-29

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

Recommendation The argument type should be "IScoracle".
Client Comment Acknowledged.

Listing 29:

```
26 function updateScoracleAddress(address _scoracleAddress)
   ↪ external;
```

3.30 CVF-30

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

Recommendation The return type should be "IScoracle".
Client Comment Acknowledged.

Listing 30:

```
36 function getScoracleAddress() external view returns (address);
```

3.31 CVF-31

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

Description We did not review these files
Client Comment Acknowledged.

Listing 31:

```
5 import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
  import "@openzeppelin/contracts/utils/Strings.sol";

8 import "@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol";
  import "@openzeppelin/contracts-upgradeable/access/
    ↪ OwnableUpgradeable.sol";
```

3.32 CVF-32

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

Description Defining a top-level structure in a file named after a contract makes it harder to navigate the code.
Recommendation Consider either moving the structure definition into the contract or moving all top-level types to a separate file named "types.sol" or something like this.
Client Comment Fixed in commit f4b0b98.

Listing 32:

```
11 struct NFCData {
```

3.33 CVF-33

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

Description The semantics of keys and values for this mapping is unclear.
Recommendation Consider documenting.
Client Comment Fixed in commit f4b0b98.

Listing 33:

```
21 mapping(address => uint256) private _reverseLookup;
```

3.34 CVF-34

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

Recommendation Consider documenting the usage of external contracts and the inability to initialize twice.
Client Comment Acknowledged.

Listing 34:

```
25 function initialize() public initializer {
```

3.35 CVF-35

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

Recommendation It would be more efficient to have an array of structs instead of two arrays, and there would be no need for length check
Client Comment Fixed in commit 75a9dae.

Listing 35:

```
42 function addAccounts(address[] memory accounts, bytes[] memory
   ↪ signatures) public {

63 function merge(uint256[] memory ids, bytes[] memory signatures)
   ↪ external {
```

3.36 CVF-36

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

Description This function doesn't scale. In case each of two NFCs already have too many account, such NFCs cannot be merged.
Recommendation Consider implementing an ability to split such merge into several transactions.
Client Comment Acknowledged.

Listing 36:

```
63 function merge(uint256[] memory ids, bytes[] memory signatures)
   ↪ external {
```

3.37 CVF-37

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

Recommendation Probably it should be checked that id is not equal to ‘mergeIntoId‘.
Client Comment Fixed in commit e110d2f.

Listing 37:

```
68 uint256 id = ids[i];
```

3.38 CVF-38

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

Recommendation ‘id‘ has been previously used for ‘ids[i]‘.
Client Comment Fixed in commit f4b0b98.

Listing 38:

```
79 delete _lookup[ids[i]];
80 _burn(ids[i]);
```

3.39 CVF-39

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** NFC.sol

Description This event is logged ids.length times.
Recommendation Take it out of the loop.
Client Comment Fixed in commit f4b0b98.

Listing 39:

```
81 emit NFCMerged(mergeIntoId, ids);
```

3.40 CVF-40

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

Description This event is emitted even if nothing changes
Client Comment Acknowledged.

Listing 40:

```
121 emit BaseURIUpdated(baseURI);
```

3.41 CVF-41

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

Recommendation Events are usually named via nouns, such as "Score", "ScoreType", "ScoreTypeDeactivation", "ChainlineNode", "ChainlinkOracle", "BaseFee", or "ChainlinkFee".
Client Comment Acknowledged.

Listing 41:

```
20 event ScoreUpdated(address indexed addressToScore, uint256
   ↪ lastUpdated, uint256 score, bytes extraData);

27 event ScoreTypeAdded(bytes32 indexed scoreTypeJobId, string
   ↪ scoreTypeName);

33 event ScoreTypeDeactivated(bytes32 indexed scoreTypeJobId);

39 event ChainlinkNodeUpdated(address indexed chainlinkNode);

45 event ChainlinkOracleUpdated(address indexed chainlinkOracle);

51 event BaseFeeUpdated(uint256 baseFee);

57 event ChainlinkFeeUpdated(uint256 chainlinkFee);
```

3.42 CVF-42

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

Recommendation The parameter types could be made more specific.
Client Comment Non-issue.

Listing 42:

```
39 event ChainlinkNodeUpdated(address indexed chainlinkNode);

45 event ChainlinkOracleUpdated(address indexed chainlinkOracle);
```

3.43 CVF-43

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

Recommendation The argument types could be made more specific.
Client Comment Non-issue.

Listing 43:

```
73 function updateChainlinkNode(address chainlinkNode) external;

75 function updateChainlinkOracle(address chainlinkOracle) external
   ↪ ;
```

3.44 CVF-44

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

Description The semantics of the returned values is unclear.
Recommendation Consider giving descriptive names to the returned values and/or adding a documentation comment.
Client Comment Fixed in commit f4b0b98.

Listing 44:

```
95 bool,
   string memory,
   bytes32
```

3.45 CVF-45

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

Recommendation The return types could be made more specific.
Client Comment Non-issue.

Listing 45:

```
100 function getChainlinkNode() external view returns (address);

102 function getChainlinkOracle() external view returns (address);
```

3.46 CVF-46

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

Description Specifying a particular compiler version makes is harder to migrate to new versions.
Recommendation Consider specifying as "^0.8.0".
Client Comment I prefer explicitly stating solidity versions for two reasons: 1. Consistency between team members 2. Easier to verify contracts on etherscan

Listing 46:

```
2 pragma solidity 0.8.11;
```

3.47 CVF-47

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

Recommendation Events are usually named via nouns, such as "NFC", "Account", "Merge", "BaseURI", or "CustomURI".
Client Comment Acknowledged.

Listing 47:

```
5 event NFCCreated(uint256 indexed id, address[] accounts);
  event AccountsAdded(uint256 indexed id, address[] accounts);
  event NFCMerged(uint256 indexed id, uint256[] ids);
  event BaseURIUpdated(string uri);
  event CustomURIUpdated(uint256 indexed id, string uri);
```

3.48 CVF-48

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

Recommendation It would be more efficient to pass a single array of structs with two fields rather than two parallel arrays.
Client Comment Fixed in commit 75a9dae.

Listing 48:

```
13 function addAccounts(address[] memory accounts, bytes[] memory
   ↪ signatures) external;

15 function mintAndAdd(address[] memory accounts, bytes[] memory
   ↪ signatures) external returns (uint256 id);

17 function merge(uint256[] memory ids, bytes[] memory signatures)
   ↪ external;
```

3.49 CVF-49

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

Recommendation This function should be named "getAccounts" for consistency.
Client Comment Fixed in commit f4b0b98.

Listing 49:

```
19 function getAddresses(uint256 id) external view returns (address
   ↪ [] memory);
```