ABDK CONSULTING

SMART CONTRACT AUDIT

xToken

Origination

Solidity

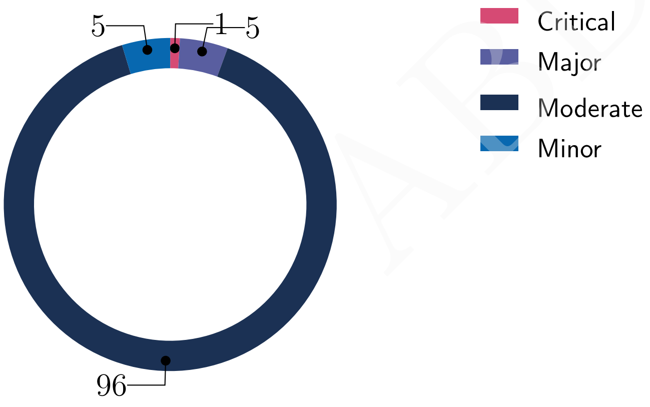
abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich

1st July 2022

We've been asked to review 21 files in a Github repository. We found 1 critical, 5 major, and a few less important issues. All identified critical and major issues have been fixed or otherwise addressed in collaboration with the client.



Findings

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

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

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

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Fixed
CVF-89	Minor	Procedural	Info
CVF-90	Minor	Suboptimal	Info
CVF-91	Minor	Suboptimal	Info
CVF-92	Minor	Procedural	Fixed
CVF-93	Minor	Bad datatype	Info
CVF-94	Minor	Bad datatype	Info
CVF-95	Minor	Bad datatype	Info
CVF-96	Minor	Bad datatype	Info
CVF-97	Minor	Procedural	Info
CVF-98	Minor	Bad datatype	Info
CVF-99	Minor	Bad datatype	Info
CVF-100	Minor	Bad datatype	Info
CVF-101	Minor	Documentation	Fixed
CVF-102	Minor	Documentation	Fixed
CVF-103	Minor	Bad datatype	Info
CVF-104	Minor	Documentation	Fixed
CVF-105	Minor	Bad datatype	Info
CVF-106	Minor	Documentation	Fixed
CVF-107	Minor	Documentation	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	June 30, 2022	D. Khovratovich	Initial Draft
0.2	June 30, 2022	D. Khovratovich	Minor revision
1.0	July 1, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

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

- interface/IFungibleOriginationPool.sol
- interface/INFTDeployer.sol
- interface/INonFungibleOriginationPool.sol
- interface/INonFungibleToken.sol
- interface/IOriginationCore.sol
- interface/IOriginationProxyAdmin.sol
- interface/IPoolDeployer.sol
- interface/IVestingEntryNFT.sol
- interface/IxTokenManager.sol
- proxies/FungibleOriginationPoolProxy.sol
- proxies/NonFungibleOriginationPoolProxy.sol
- proxies/OriginationCoreProxy.sol
- proxies/TransparentUpgradeableProxy.sol
- proxies/VestingEntryNFTProxy.sol
- FungibleOriginationPool.sol
- NFTDeployer.sol
- NonFungibleOriginationPool.sol
- OriginationCore.sol
- OriginationProxyAdmin.sol
- PoolDeployer.sol
- VestingEntryNFT.sol

The fixes were provided in the following pull requests:

- pull 1
- pull 2

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
- **Category** Procedural
- **Status** Fixed
- **Source** OriginationCore.sol

Recommendation Specifying a particular version makes it harder to migrate to newer versions. Consider specifying as "^0.8.0". Also relevant for next files: NonFungibleOriginationPool.sol, PoolDeployer.sol, OriginationProxyAdmin.sol, NFTDeployer.sol, FungibleOriginationPool.sol, IxTokenManager.sol, IVestingEntryNFT.sol, IPoolDeployer.sol, IOriginationProxyAdmin.sol, INonFungibleToken.sol, INonFungibleOriginationPool.sol, INFTDeployer.sol, IOriginationCore.sol, IFungibleOriginationPool.sol.
Client Comment There is a dependency: * @openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol (^0.8.2) limiting the minimum required compiler version, so the minimum pragma should be ^0.8.2

Listing 1:

```
pragma solidity 0.8.4;
```

3.2 CVF-2

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

Recommendation It would be more efficient to merge these two mappings into a single mapping whose keys are addresses and values are structs of two fields encapsulating the values of the original mappings. Also, this would allow fitting both, a fee amount and a fee enabled flag into a single word.
Client Comment Decided to leave it as it is.

Listing 2:

```
mapping(address => uint256) public customListingFee;

mapping(address => bool) public customListingFeeEnabled;
```

3.3 CVF-3

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

Recommendation Events are usually named via nouns, such as "FungibleListing", "NonFungibleListing", or 'ListingFee".
Client Comment Decided to leave it as it is.

Listing 3:

```
event CreateFungibleListing(address indexed pool, address indexed owner);
event CreateNonFungibleListing(address indexed pool, address indexed owner);
event SetListingFee(uint256 fee);
```

3.4 CVF-4

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment There is a comment above explaining why the constructor is there.

Listing 4:

```
constructor() initializer {}
```

3.5 CVF-5

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

Recommendation The "1e18" value should be a named constant.
Client Comment Decided to leave it as it is.

Listing 5:

```
require(_originationFee <= 1e18, "Invalid origination fee");
```

3.6 CVF-6

- **Severity** Minor
- **Category** Readability
- **Status** Info
- **Source** OriginationCore.sol

Recommendation This assignment should be made in an "else" branch of the conditional statement below.
Client Comment Decided to leave it as it is.

Listing 6:

```
address vestingEntryNFT = address(0);
```

3.7 CVF-7

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

Description This event is emitted even if nothing actually changes.
Client Comment Decided to leave it as it is.

Listing 7:

```
emit SetListingFee(_listingFee);
```

3.8 CVF-8

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

Description These functions should log some events.
Client Comment Added events

Listing 8:

```
221 function enableCustomListingFee(address deployer, uint256
     ↪ feeAmount)

237 function disableCustomListingFee(address deployer) public
     ↪ onlyOwner {
```

3.9 CVF-9

- **Severity** Moderate
- **Category** Flaw
- **Status** Fixed
- **Source** OriginationCore.sol

Description This check doesn't guarantee that a custom listing fee doesn't exceed the default listing fee, as the default listing fee could be decreased after setting a custom listing fee.
Recommendation Consider removing this check, and using the minimum of the custom listing fee for a user and the default listing fee.
Client Comment Removed the check

Listing 9:

```
225 require(
        feeAmount < listingFee,
        "Custom fee should be less than flat deployment fee"
    );
```

3.10 CVF-10

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

Recommendation The argument type should be "IERC20".
Client Comment Decided to leave it as it is.

Listing 10:

```
246 function claimFees(address _feeToken) external {
```

3.11 CVF-11

- **Severity** Moderate
- **Category** Flaw
- **Status** Fixed
- **Source** OriginationCore.sol

Description The returned value is ignored.
Recommendation Consider requiring the returned value to be "true".
Client Comment Fixed by requiring the returned value to be true

Listing 11:

```
258 IERC20(_feeToken).transfer(
```

3.12 CVF-12

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

Description The role of this constant is unclear.
Recommendation Consider documenting.
Client Comment Removed constant since it is no longer used.

Listing 12:

```
29 uint256 constant TIME_PRECISION = 1e10;
```

3.13 CVF-13

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

Description The number format of these variables is unclear.
Recommendation Consider documenting.
Client Comment Added documentation in the interface.

Listing 13:

```
48 uint256 public publicStartingPrice;

50 uint256 public publicEndingPrice;

52 uint256 public whitelistStartingPrice;

54 uint256 public whitelistEndingPrice;
```

3.14 CVF-14

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

Recommendation It should be added whether these timestamps are included into the actual sale period.
Client Comment I think the comment is clear enough.

Listing 14:

```
66 // the timestamp of the beginning of the sale

68 // the timestamp of the end of the sale
```

3.15 CVF-15

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

Recommendation Events are usually named via nouns, such as "SaleInitiation" or "MInt".
Client Comment Decided to leave it as it is.

Listing 15:

```
89 event InitiateSale(uint256 saleInitiatedTimestamp);
90 event Minted(
```

3.16 CVF-16

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment There is a comment above explaining why the constructor is there.

Listing 16:

```
111 constructor() initializer {}
```

3.17 CVF-17

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

Description There is no range check for this argument.
Recommendation Consider adding appropriate checks.
Client Comment There is a range check in OriginationCore.

Listing 17:

```
uint256 _originationFee,
```

3.18 CVF-18

- **Severity** Major
- **Category** Flaw
- **Status** Info
- **Source** NonFungibleOriginationPool.sol

Description These checks revert in case the quantity to mint is too large. This could make it hard to mint exactly the remaining quantity, as another user may front run the transaction minting a small amount, thus reducing the remaining quantity and making the original transaction to revert.
Recommendation Consider reducing the quantity to be minted instead of reverting.
Client Comment This could cause unintended behaviour when interacting with the contracts. Decided to leave it as it is for UX reasons.

Listing 18:

```
require(
    userMints[sender] + _quantityToMint <=
        maxMintablePerWhitelistedAddress,
    "User mint cap reached"
);

require(
    whitelistMints + _quantityToMint <= maxWhitelistMintable,
    "Exceeds whitelist supply"
);

require(
    totalMints + _quantityToMint <= maxTotalMintable,
    "Total mint cap reached"
);
```

3.19 CVF-19

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

Recommendation 1e18 should be a named constant.
Client Comment Decided to leave it as it is.

Listing 19:

```
207 uint256 fee = (totalCost * originationFee) / 1e18;
```

3.20 CVF-20

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

Description These functions should emit some events.
Client Comment Added the events

Listing 20:

```
245 function setWhitelist(bytes32 _whitelistMerkleRoot)

310 function setManager(address _manager) external onlyOwner {
```

3.21 CVF-21

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

Description This function is overcomplicated.
Recommendation It could be simplified using the following formula: mintPrice = (periodStartingPrice * (saleDuration - timeElapsed) + periodEndingPrice * timeElapsed) / saleDuration;
Client Comment Implemented recommendation.

Listing 21:

```
318 function getCurrentMintPrice() public view returns (uint256
      ↪ mintPrice) {
```

3.22 CVF-22

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

Description Checks like these in "view" functions look weird, as they don't allow using the functions after certain time.
Recommendation Consider removing these checks and modifying the functions to always return reasonable values.
Client Comment Decided to leave it as it is.

Listing 22:

```
require(
    isWhitelistMintPeriod() || isPublicMintPeriod(),
    "Inactive sale"
);
```

3.23 CVF-23

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

Recommendation The type of this variable should be "IFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 23:

```
address public fungibleOriginationPoolImplementation;
```

3.24 CVF-24

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

Recommendation The type of this variable should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 24:

```
address public nonFungibleOriginationPoolImplementation;
```

3.25 CVF-25

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

Recommendation The type of this argument should be "IFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 25:

```
address _fungibleOriginationPoolImplementation ,

address _fungibleOriginationPoolImplementation
```

3.26 CVF-26

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

Recommendation The type of this argument should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 26:

```
address _nonFungibleOriginationPoolImplementation

address _nonFungibleOriginationPoolImplementation
```

3.27 CVF-27

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

Recommendation The return type should be "IFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 27:

```
returns (address pool)
```

3.28 CVF-28

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

Recommendation The return type should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 28:

```
returns (address pool)
```

3.29 CVF-29

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

Description These events are emitted even if nothing actually changed.
Client Comment Decided to leave it as it is.

Listing 29:

```
emit FungibleOriginationPoolImplementationSet(

emit NonFungibleOriginationPoolImplementationSet(
```

3.30 CVF-30

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

Recommendation The type of this parameter should be "IFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 30:

```
address indexed fungibleOriginationPoolImplementation
```

3.31 CVF-31

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

Recommendation The type of this parameter should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 31:

```
address indexed nonFungibleOriginationPoolImplementation
```

3.32 CVF-32

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

Recommendation The selector value could be obtained as: TransparentUpgradeableProxy.implementation.selector.
Client Comment Decided to leave it as it is.

Listing 32:

```
hex"5c60da1b"
```

3.33 CVF-33

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

Recommendation The selector value could be obtained as: TransparentUpgradeableProxy.implementation.admin.
Client Comment Decided to leave it as it is.

Listing 33:

```
hex"f851a440"
```

3.34 CVF-34

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

Recommendation This function should emit some event.
Client Comment Added ProxyOwnershipTransferred event

Listing 34:

```
function transferProxyOwnership(address proxy, address newAdmin)
```

3.35 CVF-35

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

Description Transferring ownership to a zero address is a common way to renounce ownership. This check prevents such scenario.
Recommendation Consider removing this check.
Client Comment Decided to leave it as it is.

Listing 35:

```
require(newAdmin != address(0x0), "Admin cannot be the zero
    ↪ address");
```

3.36 CVF-36

- **Severity** Minor
- **Category** Bad naming
- **Status** Fixed
- **Source** OriginationProxyAdmin.sol

Description The name is confusing as this function overwrites the current admin rather than adds a new admin.
Recommendation Consider renaming to "setProxyAdmin".
Client Comment Implemented recommendation.

Listing 36:

```
function addProxyAdmin(address proxy, address admin) external
    ↪ onlyOwner {
```

3.37 CVF-37

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

Recommendation Should be "^0.8.0" according to a common best practice. Also relevant for next files: VestingEntryNFTProxy.sol, OriginationCoreProxy.sol, NonFungibleOriginationPoolProxy.sol, FungibleOriginationPoolProxy.sol.
Client Comment There is a dependency: * @openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol (^0.8.2), so the minimum pragma should be ^0.8.2

Listing 37:

```
pragma solidity ^0.8.4;
```

3.38 CVF-38

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment Have an explanation above why the constructor is needed.

Listing 38:

```
constructor() initializer {}
```

3.39 CVF-39

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

Description Using "_mint" here rather than "_safeMint" doesn't allow the recipient, in case it is a smart contract, to react on the received token.
Recommendation Consider using "_safeMint" and setting the vesting amounts for the token before minting.
Client Comment Replaced "_mint" with "_safeMint"

Listing 39:

```
_mint(to, tokenId);
```

3.40 CVF-40

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

Recommendation Some event should be emitted here to log the vesting amounts for the new token.
Client Comment Added an event emit here

Listing 40:

```
tokenIdVestingAmounts[tokenId] = vestingAmounts;
```

3.41 CVF-41

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

Description This function should emit some event.
Client Comment Added an event

Listing 41:

```
function setVestingAmounts(
```

3.42 CVF-42

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

Recommendation The type of this variable should be "IVestingEntryNFT".
Client Comment Decided to leave it as it is.

Listing 42:

```
address public vestingEntryNFTImplementation;
```

3.43 CVF-43

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

Recommendation The argument type should be "IVestingEntryNFT".
Client Comment Decided to leave it as it is.

Listing 43:

```
constructor(address _vestingEntryNFTImplementation) {
```

3.44 CVF-44

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

Recommendation The return type should be "IVestingEntryNFT".
Client Comment Decided to leave it as it is.

Listing 44:

```
function deployVestingEntryNFT() external returns (address pool)
    ↪ {
```

3.45 CVF-45

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

Description The role of this constant is unclear.
Recommendation Consider documenting.
Client Comment Removed constant since it is no longer used.

Listing 45:

```
uint256 constant TIME_PRECISION = 1e10;
```

3.46 CVF-46

- **Severity** Minor
- **Category** Bad naming
- **Status** Fixed
- **Source** FungibleOriginationPool.sol

Description These names are confusing. In ERC-20 the term "decimals" is used for the number of decimals in token amounts, while here it is used for the amount of base units in a single full token.
Recommendation Consider renaming to "offerTokenUnit" and "purchaseTokenUnit" respectively.
Client Comment Implemented recommendation.

Listing 46:

```
uint256 private offerTokenDecimals;

uint256 private purchaseTokenDecimals;
```

3.47 CVF-47

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

Description The number format of these variables is unclear.
Recommendation Consider documenting.
Client Comment Documented in interface.

Listing 47:

```
uint256 public publicStartingPrice;

uint256 public publicEndingPrice;

uint256 public whitelistStartingPrice;

uint256 public whitelistEndingPrice;
```

3.48 CVF-48

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

Recommendation It would be more efficient to merge these the mappings into a single mapping whose keys are addresses and values are structs encapsulating the values of the original mappings.
Client Comment Decided to leave it as it is.

Listing 48:

```
mapping(address => uint256) public userToVestingId;

mapping(address => uint256) public offerTokenAmountPurchased;

mapping(address => uint256) public purchaseTokenContribution;
```

3.49 CVF-49

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

Recommendation Events are usually named via nouns, such as "SaleInitiation".
Client Comment Decided to leave it as it is.

Listing 49:

```
event InitiateSale(uint256 totalOfferingAmount);

event CreateVestingEntry(

event ClaimVested(
```

3.50 CVF-50

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

Recommendation It is a good parctice to put a comment into an empty block to explain why the block is empty.
Client Comment There is a comment above explaining why the constructor is there.

Listing 50:

```
constructor() initializer {}
```

3.51 CVF-51

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

Description There is no range check for this argument.
Recommendation Consider adding appropriate checks.
Client Comment Pools are deployed and initialized only from OriginationCore, where the range checks are implemented.

Listing 51:

```
uint256 _originationFee,
```

3.52 CVF-52

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

Recommendation The type of this argument should be "VestingEntryNFT".
Client Comment Decided to leave it as it is.

Listing 52:

```
address _vestingEntryNFT,
```

3.53 CVF-53

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

Description In ERC-20 the "decimals" property is used by UI to render token amounts in a human-friendly way. Using this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.
Client Comment This is used to convert between different token decimal representations

Listing 53:

```
offerTokenDecimals = 10**offerToken.decimals();

       : 10**purchaseToken.decimals();
```

3.54 CVF-54

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

Description The "saleDuration" value calculated here could be higher than the "MAX_SALE_DURATION" value. This is confusing.
Recommendation Consider either renaming the constant or the variable or fixing this inconsistency in some other way.
Client Comment Decided to leave it as it is.

Listing 54:

```
saleDuration = whitelistSaleDuration + publicSaleDuration;
```

3.55 CVF-55

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

Description Usually a contract is initialized when deployed, but here a contract deployed in the past is initialized.
Recommendation Consider creating a proxy contract here just before initializing it.
Client Comment Actually the proxy contract is created in the method 'createFungibleListing' in OriginationCore. In the same function, the initialize method is called. The goal here is separation of concerns, since the FungiblePool doesn't need to be able to deploy NFT proxies. All of the deployment and initialization logic is contained in OriginationCore.

Listing 55:

```
206 vestingEntryNFT.initialize("VestingNFT", "VNFT", address(this));
```

3.56 CVF-56

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

Description The expression "purchaseTokenContribution[msg.sender]" is calculated twice.
Recommendation Consider calculating once and reusing.
Client Comment Implemented recommendation.

Listing 56:

```
242 purchaseTokenContribution[msg.sender] + contributionAmount >

247     purchaseTokenContribution[msg.sender];
```

3.57 CVF-57

- **Severity** Major
- **Status** Info
- **Category** Flaw
- **Source** FungibleOriginationPool.sol

Description Already purchased contributions are tracked per message sender, while there could be several leafs in the merkle tree for the same message sender. This would not allow a user to fully use all his leafs.
Recommendation Consider tracking purchased contributions per merkle tree leaf.
Client Comment There will be only one leaf per address in the merkle tree.

Listing 57:

```
242 purchaseTokenContribution[msg.sender] + contributionAmount >
```

3.58 CVF-58

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

Description This check forbids zero amount purchases only when the specified contribution amount exceeds the remaining allowed contribution amount. It is still possible to perform a zero amount purchase by explicitly specifying zero as "contributionAmount".
Recommendation Consider moving this check after the conditional statement to forbid zero amount purchases.
Client Comment There is already a min contribution amount check in _purchase.

Listing 58:

```
require(
    contributionAmount != 0,
    "User has reached his max contribution amount"
);
```

3.59 CVF-59

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

Recommendation This variable is redundant, as "msg.sender" is cheaper to access than a local variable.
Client Comment Implemented recommendation.

Listing 59:

```
address sender = msg.sender;
```

3.60 CVF-60

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Fixed
- **Source** FungibleOriginationPool.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.
Recommendation Consider using the "muldiv" function as described here: https://2π.com/21/muldiv/index.html or some other approach that prevents phantom overflow.
Client Comment Implemented recommendation.

Listing 60:

```
uint256 feeInPurchaseToken = _divUp(
    contributionAmount * originationFee,
    1e18
);

offerTokenAmount = _divUp(
    _divUp(
        contributionAmount * purchaseTokenDecimals,
        offerTokenPrice
    ) * offerTokenDecimals,
    purchaseTokenDecimals
);

    : ((timeSinceInit * tokenAmount) / vestingPeriod) -
```

3.61 CVF-61

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

Recommendation This assignment should be made after the conditional statement below to use the potentially adjusted contribution amount.
Client Comment Since this case can be hit only once in the sale's duration in the case of full sell out and may not be hit at all, decided to leave it as it is.

Listing 61:

```
uint256 feeInPurchaseToken = _divUp(
    contributionAmount * originationFee,
    1e18
);
```

3.62 CVF-62

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

Recommendation The value "1e18" should be a named constant.
Client Comment Decided to leave it as it is.

Listing 62:

```
1e18

          1e18
```

3.63 CVF-63

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

Description The expression "offertokenAmountSold + offerTokenAmount" is calculated twice.
Recommendation Consider calculating once and reusing.
Client Comment Since this case can be hit only once in the sale's duration in the case of full sell out and may not be hit at all, decided to leave it as it is.

Listing 63:

```
if (offerTokenAmountSold + offerTokenAmount >
   ↪ totalOfferingAmount) {

    uint256 refundAmountInOfferTokens = offerTokenAmountSold +
        offerTokenAmount -
```

3.64 CVF-64

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

Recommendation In case of an ERC-20 purchase token, it would be more efficient to just transfer less tokens from the user instead of sending some tokens back to him.
Client Comment This transaction is only at the end of a token sale and only if the token sale is finalized. Decided to not change the code, since it doesn't have that big of an impact.

Listing 64:

```
_returnPurchaseTokens(msg.sender, refundAmount);
```

3.65 CVF-65

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

Recommendation The "refundAmount" value was already subtracted from "contributionAmount" a few lines above and it shouldn't be subtracted again here.

Listing 65:

```
contributionAmount − refundAmount,
```

3.66 CVF-66

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

Recommendation The numerator should be multiplied by "originationFee" before dividing by "1e18".
Client Comment Fixed by multiplying by "originationFee"

Listing 66:

```
contributionAmount − refundAmount,
```

3.67 CVF-67

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

Description This may never be true, as the logic above doesn't allow purchasing purchasing more than the remaining offer amount.
Recommendation Consider removing this check or turning it into an assert to emphasize that it should never be violated.
Client Comment Decided to leave it as it is since the check can be true if ‘offerTokenAmountSold == totalOfferingAmount‘

Listing 67:

```
offerTokenAmountSold <= totalOfferingAmount,
```

3.68 CVF-68

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

Description In case a user made several purchases, each subsequent purchase will overwrite the previous value here making the "userToVestingId" mapping useless.
Recommendation Consider removing this mapping.
Client Comment This mapping is used for testing mostly, will have a separate subgraph to track the vesting entries for an user.

Listing 68:

```
userToVestingId[_sender] = vestingID;
```

3.69 CVF-69

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

Description This check makes the contract less convenient to use, as a user has to track what NFTs are fully claimed and remove them from the list.
Recommendation Consider removing this check.
Client Comment This will be tracked in the frontend.

Listing 69:

```
require(
    tokenAmount != tokenAmountClaimed,
    "User has already claimed his token vesting"
);
```

3.70 CVF-70

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

Description This should be done only when "offerTokenPayout" is not zero.
Client Comment Decided to leave it as it is.

Listing 70:

```
offerToken.safeTransfer(msg.sender, offerTokenPayout);
```

3.71 CVF-71

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

Description Checks like these in "view" functions look weird, as they don't allow using the functions after certain time.
Recommendation Consider removing these checks and modifying the functions to always return reasonable values.
Client Comment Removed the checks and return the default starting price in case the sale hasn't started or is over.

Listing 71:

```
485 require(
        block.timestamp <= saleEndTimestamp,
        "Sale not started or over"
    );

514 require(
        block.timestamp <= saleEndTimestamp,
        "Sale not started or over"
    );
```

3.72 CVF-72

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

Description This function is overcomplicated.
Recommendation It could be simplified using the following formula: offerTokenPrice = (_startingPrice * (saleDuration - timeElapsed) + _endingPrice * timeElapsed) / saleDuration;
Client Comment Implemented the recommendation.

Listing 72:

```
531 function getOfferTokenPrice()
```

3.73 CVF-73

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

Recommendation Brackets are redundant here.
Client Comment Decided to leave it as it is for clarity

Listing 73:

```
: ((timeSinceInit * tokenAmount) / vestingPeriod) −
```

3.74 CVF-74

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

Description This check is redundant as it checks what is already checked a few lines above.
Recommendation Consider removing this check.
Client Comment Fixed by removing the check.

Listing 74:

```
require(success);
```

3.75 CVF-75

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

Description These functions should emit some events.
Client Comment Added the events.

Listing 75:

```
function setWhitelist(bytes32 _whitelistMerkleRoot)

function setManager(address _manager) external onlyOwner {
```

3.76 CVF-76

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** FungibleOriginationPool.sol

Description Phantom overflow is possible here, while division rounded up should never overflow.
Recommendation Consider implementing like: return num / div + (num % div == 0 ? 0 : 1);
Client Comment Implemented the recommendation.

Listing 76:

```
705 return (num + div - 1) / div;
```

3.77 CVF-77

- **Severity** Major
- **Category** Procedural
- **Status** Fixed
- **Source** OriginationCoreProxy.sol

Description The contract with the same name exists in this code base. Having two contracts with the same name in one project makes code harder to read and more error prone.
Recommendation Consider either using the local contract or renaming the local contract to make in distinguishable from the OpenZeppelin's contract.
Client Comment Fixed by using local contract

Listing 77:

```
4 import "@openzeppelin/contracts/proxy/transparent/
    ↪ TransparentUpgradeableProxy.sol";
```

3.78 CVF-78

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

Description It is not checked that ""_logic"" equals "_poolDeployer.nonFungibleOriginationPoolImplementation()".
Recommendation Consider adding such check. Alternatively, consider obtaining the current implementation address from the pool deployer.
Client Comment Pool deployments are made only through PoolDeployer, which deploys pools pointing to the implementations, so the logic will always be equal to "_poolDeployer.nonFungibleOriginationPoolImplementation()"

Listing 78:

```
address _logic ,
```

3.79 CVF-79

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

Recommendation The type of this argument should be "IPoolDeployer".
Client Comment Decided to leave it as it is.

Listing 79:

```
address _poolDeployer
```

3.80 CVF-80

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

Description This check makes the "_implementation" argument redundant, as its value could be obtained from the pool deployer.
Recommendation Consider removing the argument.
Client Comment Decided to leave it as it is to keep compatibility with OpenZeppelin's TransparentUpgradeableProxy interface.

Listing 80:

```
poolDeployer.nonFungibleOriginationPoolImplementation() ==
      _implementation,
"Can only upgrade to latest nonFungibleOriginationPool
   ↪ implementation"

poolDeployer.nonFungibleOriginationPoolImplementation() ==
      _implementation,
"Can only upgrade to latest nonFungibleOriginationPool
   ↪ implementation"
```

3.81 CVF-81

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

Recommendation This code could be simplified by using the "_upgradeToAndCall" function here.
Client Comment Fixed by implementing the recommendation.

Listing 81:

```
_upgradeTo(_implementation);
Address.functionDelegateCall(_implementation, data);
```

3.82 CVF-82

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

Description Testing the value of a constant looks like waste of gas.
Recommendation Consider removing this check.
Client Comment The code is actually the same as OpenZeppelin's TransparentUpgradeableProxy. Decided to leave it as it is.

Listing 82:

```
assert(
    _ADMIN_SLOT ==
        bytes32(uint256(keccak256("eip1967.proxy.admin")) - 1)
);
```

3.83 CVF-83

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

Recommendation It would be more efficient to use the "_upgradeTo" function here.
Client Comment The code is actually the same as OpenZeppelin's TransparentUpgradeableProxy. Decided to leave it as it is.

Listing 83:

```
_upgradeToAndCall(newImplementation, bytes(""), false);
```

3.84 CVF-84

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

Description This check consumes some gas. Is it really necessary?
Client Comment The code is actually the same as OpenZeppelin's TransparentUpgradeableProxy. Decided to leave it as it is.

Listing 84:

```
msg.sender != _getAdmin(),
```

3.85 CVF-85

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

Description It is not checked that "_logic" equals "_poolDeployer.fungibleOriginationPoolImplementation()".
Recommendation Consider adding such check. Alternatively, consider obtaining the current implementation address from the pool deployer.
Client Comment Pool deployments are made only through PoolDeployer, which deploys pools pointing to the implementations, so the logic will always be equal to "_poolDeployer.fungibleOriginationPoolImplementation()"

Listing 85:

```
address _logic,
```

3.86 CVF-86

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

Recommendation The type of this argument should be "IPoolDeployer".
Client Comment Decided to leave it as it is.

Listing 86:

```
address _poolDeployer
```

3.87 CVF-87

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

Description This check makes the "_implementation" argument redundant, as its value could be obtained from the pool deployer.
Recommendation Consider removing the argument.
Client Comment Decided to leave it as it is to keep compatibility with OpenZeppelin's TransparentUpgradeableProxy interface.

Listing 87:

```
poolDeployer.fungibleOriginationPoolImplementation() ==
    _implementation,

poolDeployer.fungibleOriginationPoolImplementation() ==
    _implementation,
```

3.88 CVF-88

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

Recommendation This code could be simplified by using the "_upgradeToAndCall" function here.
Client Comment Fixed using recommendation.

Listing 88:

```
_upgradeTo(_implementation);
Address.functionDelegateCall(_implementation, data);
```

3.89 CVF-89

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

Description This interface is not used.
Recommendation Consider removing it.
Client Comment The interface is used in line 244 of OriginationCore.sol, to check if the caller of ‘claimFees‘ is the RevenueController.

Listing 89:

```
interface IxTokenManager {
```

3.90 CVF-90

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

Description These functions should be emit some events and these events should be declared in this interface.
Client Comment The functions don’t emit events in the implementation.

Listing 90:

```
function addManager(address manager, address fund) external;

function setRevenueController(address controller) external;
```

3.91 CVF-91

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

Description There are functions to check whether an address is the manager of a fund or the revenue controller, but there are no functions to get the current manager of a fund or the current revenue controller.
Recommendation Consider adding such functions.
Client Comment There can be multiple managers of a fund, and there is no need to get the current revenue controller because it can’t be changed.

Listing 91:

```
function isManager(address manager, address fund)

function isRevenueController(address caller) external view
  ↪ returns (bool);
```

3.92 CVF-92

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

Recommendation This import is not used, consider removing it.
Client Comment Removed the import.

Listing 92:

```
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
```

3.93 CVF-93

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

Recommendation The return type should be "IFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 93:

```
8  returns (address);

17 returns (address pool);
```

3.94 CVF-94

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

Recommendation The return type should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 94:

```
13 returns (address);

21 returns (address token);
```

3.95 CVF-95

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

Recommendation The argument type should be "iFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 95:

```
24 address _fungibleOriginationPoolImplementation
```

3.96 CVF-96

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

Recommendation The argument type should be "INonFungibleOriginationPool".
Client Comment Decided to leave it as it is.

Listing 96:

```
28 address _nonFungibleOriginationPoolImplementation
```

3.97 CVF-97

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

Description These functions are the same in the "IOriginationProxyAdmin" interface.
Recommendation Consider extracting into a shared base interface.
Client Comment Decided to leave it as it is.

Listing 97:

```
31 function owner() external view returns (address);

33 function renounceOwnership() external;

35 function transferOwnership(address newOwner) external;
```

3.98 CVF-98

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

Recommendation The argument type could be more specific.
Client Comment Decided to leave it as it is.

Listing 98:

```
function setOriginationInstance(address instance) external;
```

3.99 CVF-99

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

Recommendation The type of this field should be 'IERC721".
Client Comment Decided to leave it as it is.

Listing 99:

```
address collection;
```

3.100 CVF-100

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

Recommendation The type of this field should be 'IERC20".
Client Comment Decided to leave it as it is.

Listing 100:

```
address purchaseToken;
```

3.101 CVF-101

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

Description The number format of these fields is unclear.
Recommendation Consider documenting.
Client Comment Added documentation

Listing 101:

```
uint256 publicStartingPrice;

uint256 publicEndingPrice;

uint256 whitelistStartingPrice;

uint256 whitelistEndingPrice;
```

3.102 CVF-102

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

Description The number format for this argument is unclear.
Recommendation Consider documenting.
Client Comment Added documentation

Listing 102:

```
uint256 originationFee,
```

3.103 CVF-103

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

Recommendation The return type should be "IVestingEntryNFT".
Client Comment Decided to leave it as it is.

Listing 103:

```
function vestingEntryNFTImplementation() external view returns (
    ↪ address);

function deployVestingEntryNFT() external returns (address nft);
```

3.104 CVF-104

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

Description The semantics of this function is unclear.
Recommendation Consider documenting.
Client Comment Added documentation

Listing 104:

```
function receiveFees() external payable;
```

3.105 CVF-105

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

Recommendation The type of these fields should be "IERC20".
Client Comment Decided to leave it as it is since different implementations can use different IERC20 interfaces (IERC20, IERC20Metadata, etc.)

Listing 105:

```
address offerToken; // the token being offered for sale
address purchaseToken; // the token used to purchase the offered
    ↪ token
```

3.106 CVF-106

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

Description The number format of these fields is unclear.
Recommendation Consider documenting.
Client Comment Documented

Listing 106:

```
uint256 publicStartingPrice; // in purchase tokens
uint256 publicEndingPrice; // in purchase tokens
uint256 whitelistStartingPrice; // in purchase tokens
uint256 whitelistEndingPrice; // in purchase tokens
```

3.107 CVF-107

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

Description The number format of this argument is unclear.
Recommendation Consider documenting.
Client Comment Documented

Listing 107:

```
uint256 originationFee,
```