Report

v. 1.0

Customer

CMTA

Smart Contract Audit

DVP

15th December 2022

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	15.12.22	A. Zveryanskaya	Initial Draft
0.2	15.12.22	A. Zveryanskaya	Minor revision
1.0	15.12.22	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) 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.

3 Project scope

We were asked to review:

- Original Repository
- Fix Repository

Files:

contracts/
DVP.sol
/contracts/POT/
IPOT.sol

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code 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 as well as 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 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 if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure 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.

5 Our findings

We found 1 moderate, and a few less important issues. All identified issues have been fixed.

	Active	Fixed
Moderate	0	1
Minor	0	23

Fixed 24 out of 24 issues

6 Moderate Issues

CVF-13. FIXED

- **Category** Flaw
- **Source** DVP.sol

Description The returned value is ignored.

Recommendation Consider explicitly requiring the returned value to be true.

Client Comment *Proposed changes were implemented.*

```
141 IERC20Upgradeable(assetTokenAddress).transferFrom(receiver, address(
       ↪ this), numAssetTokensForSettlement);
```

```
208 IERC20Upgradeable(assetTokenAddress).transfer(sender,
       ↪ numAssetTokensForSettlement);
```

```
270 IERC20Upgradeable(assetTokenAddress).transfer(receiver,
       ↪ numAssetTokensForSettlement);
```

7 Minor Issues

CVF-1. FIXED

- **Category** Procedural
- **Source** DVP.sol

Description These imports are not used.

Client Comment *The unused imports were removed.*

```
10 import "@openzeppelin/contracts-upgradeable/token/ERC721/
      ↪ ERC721Upgradeable.sol";
   import "@openzeppelin/contracts-upgradeable/token/ERC721/
      ↪ IERC721ReceiverUpgradeable.sol";
```

CVF-2. FIXED

- **Category** Bad datatype
- **Source** DVP.sol

Description The type of this variable should be ”IPOT”.

Client Comment *Proposed changes were implemented.*

```
34 address private potAddress;
```

CVF-3. FIXED

- **Category** Bad datatype
- **Source** DVP.sol

Description The type of the ”_potAddress” argument should be ”IPOT”.

Client Comment *Proposed changes were implemented.*

```
44  function initialize(address _potAddress)
305 function setPotAddress(address _potAddress)
```

CVF-4. FIXED

- **Category** Unclear behavior
- **Source** DVP.sol

Description Unchained functions should be used here.

Client Comment *Proposed changes were implemented.*

```
51 __Pausable_init();
__Ownable_init();
__UUPSUpgradeable_init();
```

CVF-5. FIXED

- **Category** Documentation
- **Source** DVP.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Proposed changes were implemented by adding a comment inside the block.*

```
62 {}
```

CVF-6. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description All the parameters should be indexed.

Client Comment *Proposed changes were implemented.*

```
67 event DeliveryConfirmed(uint256 indexed _tokenId, address _numAt);
```

```
72 event DeliveryExecuted(uint256 indexed _tokenId, address _numAt,
   ↪ address to);
```

```
77 event SettlementCanceled(uint256 indexed _tokenId, address _numAt,
   ↪ address to);
```

CVF-7. FIXED

- **Category** Documentation
- **Source** DVP.sol

Description The semantics of the "_numAt" parameters is unclear.

Recommendation Consider documenting.

Client Comment *The variable was renamed with a clearer name instead of adding documentation.*

```
67 event DeliveryConfirmed(uint256 indexed _tokenId, address _numAt);
```

```
72 event DeliveryExecuted(uint256 indexed _tokenId, address _numAt,
   ↪ address to);
```

```
77 event SettlementCanceled(uint256 indexed _tokenId, address _numAt,
   ↪ address to);
```

CVF-8. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Several external calls to the same contract are performed in the same function. This could consume lots of gas.

Recommendation Consider implementing a single getter function in the "IPOT" interface to fetch all the needed information in one call.

Client Comment *Declare a function getDetails (replace getDetailsForSettlement) and use it to fetch all the needed information in one call.*

```
89 address owner = IPOT(potAddress).ownerOf(tokenId);
90 IPOT.potStatus potStatus = IPOT(potAddress).getStatus(tokenId);
```

```
111 address assetTokenAddress = IPOT(potAddress).getDealDetailAddress(
    ↪ tokenId);
```

```
113 address receiver = IPOT(potAddress).getReceiver(tokenId);
```

```
116 uint256 numAssetTokensForSettlement = IPOT(potAddress).
    ↪ getDealDetailNum(tokenId);
```

CVF-9. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description This should be logged at the very beginning of the function..

Client Comment *Proposed changes were implemented.*

```
93 console.log("[DVP] DVP.checkDeliveryForPot(", tokenId, ")");
```

CVF-10. FIXED

- **Category** Bad datatype
- **Source** DVP.sol

Description The asset token doesn't have to be upgradeable.

Recommendation Consider converting to "IERC20".

Client Comment *Proposed changes were implemented.*

```
114 uint256 allowance = IERC20Upgradeable(assetTokenAddress).allowance(
      ↪ receiver, address(this));

117 uint256 numAssetTokensOfReceiver = IERC20Upgradeable(
      ↪ assetTokenAddress).balanceOf(receiver);

141 IERC20Upgradeable(assetTokenAddress).transferFrom(receiver, address(
      ↪ this), numAssetTokensForSettlement);

185 uint256 numAssetTokensOfDvP = IERC20Upgradeable(assetTokenAddress).
      ↪ balanceOf(address(this));

208 IERC20Upgradeable(assetTokenAddress).transfer(sender,
      ↪ numAssetTokensForSettlement);

252 uint256 numAssetTokensOfDvP = IERC20Upgradeable(assetTokenAddress).
      ↪ balanceOf(address(this));

270 IERC20Upgradeable(assetTokenAddress).transfer(receiver,
      ↪ numAssetTokensForSettlement);
```

CVF-11. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Obtaining the allowance amount here seems redundant.

Recommendation Just try to transfer tokens and the token contract will do proper allowance check.

Client Comment *The call to get the allowance and the check of this one was removed.*

```
114 uint256 allowance = IERC20Upgradeable(assetTokenAddress).allowance(
       ↪ receiver, address(this));
```

CVF-12. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Obtaining the balance amount here seems redundant.

Recommendation Just try to transfer tokens and the token contract will do proper balance check.

Client Comment *The call to get the balance and the check of this one was removed.*

```
117 uint256 numAssetTokensOfReceiver = IERC20Upgradeable(
       ↪ assetTokenAddress).balanceOf(receiver);
```

CVF-14. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Several external calls to the same contract are performed in the same function. This could consume lots of gas.

Recommendation Consider implementing a single getter function in the "IPOT" interface to fetch all the needed information in one call.

Client Comment *Declare a function getDetailsForDelivery and use it to fetch all the needed information in one call.*

```
168 IPOT.potStatus potStatus = IPOT(potAddress).getStatus(tokenId);

177 address owner = IPOT(potAddress).ownerOf(tokenId);

184 address assetTokenAddress = IPOT(potAddress).getDealDetailAddress(
      ↪ tokenId);

187 uint256 numAssetTokensForSettlement = IPOT(potAddress).
      ↪ getDealDetailNum(tokenId);
    address sender = IPOT(potAddress).getSender(tokenId);
```

CVF-15. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Obtaining the balance amount here seems redundant.

Recommendation Just try to transfer tokens and the token contract will do proper balance check.

Client Comment *The call to get the balance and the check of this one was removed.*

```
185 uint256 numAssetTokensOfDvP = IERC20Upgradeable(assetTokenAddress).
      ↪ balanceOf(address(this));
```

CVF-16. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Several external calls to the same contract are performed in the same function. This could consume lots of gas.

Recommendation Consider implementing a single getter function in the "IPOT" interface to fetch all the needed information in one call.

Client Comment *Declare a function getDetailsForSettlement and use it to fetch all the needed information in one call.*

```
242 IPOT.potStatus potStatus = IPOT(potAddress).getStatus(tokenId);

248 address owner = IPOT(potAddress).ownerOf(tokenId);

251 address assetTokenAddress = IPOT(potAddress).getDealDetailAddress(
      ↪ tokenId);

254 uint256 numAssetTokensForSettlement = IPOT(potAddress).
      ↪ getDealDetailNum(tokenId);

269 address receiver = IPOT(potAddress).getReceiver(tokenId);
```

CVF-17. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Obtaining the balance amount here seems redundant.

Recommendation Just try to transfer tokens and the token contract will do proper balance check.

Client Comment *The call to get the balance and the check of this one was removed.*

```
252 uint256 numAssetTokensOfDvP = IERC20Upgradeable(assetTokenAddress).
      ↪ balanceOf(address(this));
```

CVF-18. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description Several external calls to the same contract are performed in the same function. This could consume lots of gas.

Recommendation Consider implementing a single getter function in the "IPOT" interface to fetch all the needed information in one call.

Client Comment *Declare a function getStatusAndMintTime and use it to fetch all the needed information in one call.*

```
287 IPOT.potStatus potStatus = IPOT(potAddress).getStatus(tokenId);
```

```
293 uint256 mintTime = IPOT(potAddress).getMintTime(tokenId);
```

CVF-19. FIXED

- **Category** Suboptimal
- **Source** DVP.sol

Description This function should emit some event.

Client Comment *Proposed changes were implemented.*

```
305 function setPotAddress(address _potAddress)
```

CVF-20. FIXED

- **Category** Bad datatype
- **Source** DVP.sol

Description The return type should be "IPOT".

Client Comment *Emit the event POTAddressChanged.*

```
318 returns (address)
```

CVF-21. FIXED

- **Category** Procedural
- **Source** IPOT.sol

Description Consider specifying as "^0.8.12" or "^0.8.0".

Client Comment *The import is fixed to the version 0.8.17 to be consistent with DVP.sol*

```
2 pragma solidity >=0.8.12 <0.9.0;
```

CVF-22. FIXED

- **Category** Procedural
- **Source** IPOT.sol

Description Should be: import "@openzeppelin/contracts/token/ERC721/IERC721.sol";

Client Comment *Proposed changes were implemented.*

```
4 import "@openzeppelin/contracts/token/ERC721/extensions/
    ↪ ERC721Pausable.sol";
```

CVF-23. FIXED

- **Category** Readability
- **Source** IPOT.sol

Description In some cases argument names are prefixed with underscore, while in other cases they are not.

Recommendation Consider using a consistent naming schema.

Client Comment *Add a prefix with underscore to all arguments.*

```
15 function initiatePayment(uint256 tokenId)
18 function deactivatePot(uint256 tokenId)
21 function getMintTime(uint256 _tokenId)
26 function getStatus(uint256 _tokenId)
31 function getSender(uint256 _tokenId)
36 function getReceiver(uint256 _tokenId)
41 function getDealDetailNum(uint256 _tokenId)
46 function getDealDetailAddress(uint256 _tokenId)
```

CVF-24. FIXED

- **Category** Procedural
- **Source** IPOT.sol

Description These functions should emit some events and these events should be declared in this interface.

Client Comment *Move the events from POT.sol into IPOT.sol*

```
15 function initiatePayment(uint256 tokenId)
18 function deactivatePot(uint256 tokenId)
```

ABDK
Consulting

About us

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.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting