Report

v. 1.0

Customer

Spectral Finance

Smart Contract Audit

SC POC

9th January 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	08.01.24	A. Zveryanskaya	Initial Draft
0.2	09.01.24	A. Zveryanskaya	Minor revision
1.0	09.01.24	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.

Spectral's Machine Intelligence Network, is a two-sided marketplace offering a cost-effective, high-quality way for smart contracts to consume machine learning (ML) and artificial intelligence (AI) inferences using zero-knowledge machine learning (zkML), a unique validation process, and an easy-to-use, state-of-the-art software development kit.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	BeaconProxy Deployer.sol	Competition.sol	CompetitionFactory.sol
	Consumption.sol	DataTypes.sol	MathHelper.sol
	Modeler.sol	ServiceAgreement.sol	SpectralToken.sol
interfaces/			
	ICompetition.sol	ICompetitionFactory.sol	IConsumption.sol
	IMathHelper.sol	IModeler.sol	IServiceAgreement.sol
libraries/			
	ModelerLibrary.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.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 2 critical, 14 major, and a few less important issues. All identified Critical and Major issues have been fixed.

Issues	Info	Fixed
Severity: Critical	0	2
Major	0	14
Moderate	1	8
Minor	19	54

Fixed 77 out of 98 issues

6 Critical Issues

CVF-1. FIXED

- **Category** Flaw
- **Source** ServiceAgreement.sol

Description This value is 19592 days, rather than 28 days.

Client Comment *Changed value to 2419200, which is 28 days in seconds*

```
26 require(block.timestamp > lastUpdate + 1692748800, "Service␣
     ↪ agreement␣can␣only␣be␣updated␣once␣every␣28␣days");
```

CVF-2. FIXED

- **Category** Overflow/Underflow
- **Source** MathHelper.sol

Description Underflow is possible here.

Recommendation Consider, for example, the following case: arr = [0, 1] left = 0; right = 1;

Client Comment *Quicksort removed from the scope after below suggestions remove its necessity.*

```
20 j--;
```

7 Major Issues

CVF-3. FIXED

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

Description This contract should inherit the ”ICompetition” interface.

Client Comment *Modified to conform to your suggestion.*

```
11 contract Competition is Initializable {
```

CVF-4. FIXED

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

Description This should be done only when i != j.

Client Comment *Omitted from scope after below suggestions removed its necessity.*

```
104 (topNModelers[i], topNModelers[j]) = (topNModelers[j], topNModelers[
      ↪ i]);
```

CVF-5. FIXED

- **Category** Suboptimal
- **Source** Competition.sol

Description Linear search is suboptimal.

Recommendation Consider accepting the old modeler index as a hint argument.

Client Comment *Modified to conform to your suggestion.*

```
121 for (uint256 i = 0; i < n; ++i) {
```

CVF-6. FIXED

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

Description This contract should inherit from ”ICompetitionFactory”.

Client Comment *This contract now inherits from ICompetitionFactory as suggested.*

```
11 contract CompetitionFactory is Initializable, UUPSUpgradeable,
   ↪ ProxyBeaconDeployer {
```

CVF-14. FIXED

- **Category** Flaw
- **Source** MathHelper.sol

Description There is no check to ensure right >= left.

Recommendation Consider adding such a check.

Client Comment *Omitted from scope after below suggestions removed its necessity.*

```
9 function quickSort(uint256[] memory arr, uint left, uint right)
   ↪ public pure {
```

CVF-15. FIXED

- **Category** Suboptimal
- **Source** MathHelper.sol

Description The swap shouldn’t be performed in case i == j.

Client Comment *Omitted from scope after below suggestions removed its necessity.*

```
17 if (i <= j) {
       (arr[uint256(i)], arr[uint256(j)]) = (arr[uint256(j)], arr[
          ↪ uint256(i)]);
```

CVF-16. FIXED

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

Description There are two cases here: ether "j" points to the pivot or it points to the element right before the pivot. In the former case "j - 1' should be used as the new right boundary.

```
24 quickSort(arr, left, j);
```

CVF-17. FIXED

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

Description There are two cases here: ether "i" points to the pivot or it points to the element right after the pivot. In the former case "i + 1' should be used as the new left boundary.

```
26 quickSort(arr, i, right);
```

CVF-18. FIXED

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

Description This contract should inherit from the "IModeler" interface.

Client Comment *The code has been modified as suggested.*

```
10 contract Modeler is Initializable, PausableUpgradeable  {
```

CVF-19. FIXED

- **Category** Flaw
- **Source** Modeler.sol

Description There is no access level specified for these variable, so internal access will be used by default.

Recommendation Consider explicitly specifying and access level.

Client Comment *Private access level added to the variables.*

```
13 uint256 MAX_VALIDATORS;
   address validatorToken;
   uint256 validatorStakeAmount;
```

```
30 address competitionContract;
   address consumptionContract;
```

CVF-20. FIXED

- **Category** Flaw
- **Source** Modeler.sol

Description There is no range check for this argument.

Recommendation Consider adding an appropriate check.

Client Comment *Check added to require max_validators > 0.*

```
66 uint256 _maxValidators
```

CVF-21. FIXED

- **Category** Suboptimal
- **Source** Modeler.sol

Recommendation This check is redundant, as it is superseded by the ”onlyValidator” modifier.

Client Comment *Omitted the redundant check.*

```
81 require(validators[msg.sender].isRegistered);
```

CVF-22. FIXED

- **Category** Suboptimal
- **Source** BeaconProxyDeployer.sol

Recommendation These flags are redundant. Just use a zero beacon address as an indicator that the beacon isn't set.

Client Comment *Replaced variable with a require that checks if the beacon is the 0 address.*

```
10 bool private isCompetitionBeaconSet;
   bool private isModelerBeaconSet;
```

CVF-23. FIXED

- **Category** Flaw
- **Source** BeaconProxyDeployer.sol

Description This event doesn't distinguish what beacon was deployed, competition or modeler.

Recommendation Consider using two separate events or introducing an additional even parameter.

Client Comment *Created two separate events.*

```
13 event BeaconDeployed(address _beacon);
```

8 Moderate Issues

CVF-24. FIXED

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

Description This allows calling this function again for the same modeler before 1 day interval passed from the previous call.

Recommendation Consider either forbidding this or explaining why this is fine.

Client Comment *Modified the function to add an additional require to check if they're currently in the 24 hour opting out window.*

```
54 require(modeler.optOutTime > block.timestamp, "Modeler␣already␣opted
   ↪ ␣out");
```

```
56 modeler.optOutTime = block.timestamp + 1 days; //modeler must be
   ↪ online for the next 24 hours or they may be slashed
```

CVF-25. INFO

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

Description This function allows the same validator to opt in several times.

Recommendation Consider either forbidding this or clearly explaining why this is fine.

Client Comment *There would be no incentive to call this function multiple times, and once the validator had opted in, it wouldn't change any blockchain state for them to call this function again.*

```
71 function optInValidator(DataTypes.ValidatorData storage validator,
   ↪ uint256 validatorStakeAmount, address[] storage
   ↪ validatorAddresses) external {
```

CVF-26. FIXED

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

Description This function allows the same validator to opt out several times.

Recommendation Consider either forbidding this or clearly explaining why this is fine.

Client Comment *Modified the function to add an additional require to check if they're currently in the 24 hour opting out window.*

```
79 function optOutValidator(DataTypes.ValidatorData storage validator,
   ↪ address[] storage validatorAddresses, address
   ↪ competitionContract) external{
```

```
88 function emergencyOptOutValidator(address validatorAddress,
   ↪ DataTypes.ValidatorData storage validatorData, address[]
   ↪ storage validatorAddresses) external {
```

CVF-27. FIXED

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

Description This function allows a non-registered validator to opt-out.

Recommendation Consider forbidding this.

Client Comment *Added a check to forbid non-validators calling this function.*

```
79 function optOutValidator(DataTypes.ValidatorData storage validator,
   ↪ address[] storage validatorAddresses, address
   ↪ competitionContract) external{
```

CVF-28. FIXED

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

Description This could transfer tokens to a zero address, in case the modeler contract wasn't set yet.

Recommendation Consider reverting in such a case.

Client Comment *Added a zero check in the function.*

```
69 IERC20(serviceAgreement.stakedToken()).safeTransferFrom(msg.sender,
      ↪ modelerContract, serviceAgreement.stakedAmount());
```

CVF-29. FIXED

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

Description There is no check to ensure that the new top modeler is actually new, i.e. it doesn't exist in the "topNModelers" array.

Recommendation Consider performing such a check.

Client Comment *Added a require statement in insertNewModeler to check if the modeler is already in the modelerToMedian mapping.*

```
75 function setModelerNetPerformanceResultAndUpdate(address modeler,
      ↪ uint256 medianPerformanceResults) external onlyModelerContract
      ↪  {
```

```
119 function replaceOptOutTopNModeler(address _oldModeler, address
      ↪ _newModeler, uint256 _medianPerformanceResults) external
      ↪ onlyModelerContract {
```

CVF-30. FIXED

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

Description Performing full sorting of an in-storage array after replacing or inserting a single element is an overkill.

Recommendation Consider just finding the proper insert position for the element using binary search, and then shifting the remaining array elements.

Client Comment *Amended the code with this suggestion.*

```
84          quickSort(0, topNModelers.length - 1);
```

```
116 quickSort(0, topNModelers.length - 1);
```

```
129 quickSort(0, topNModelers.length - 1);
```

CVF-33. FIXED

- **Category** Flaw
- **Source** MathHelper.sol

Description This function is declared as ”public” which means it could be called externally. However, once called externally, this function doesn’t return the sorted array, so it effectively does nothing.

Recommendation Consider declaring this function as ”internal”.

Client Comment *Code has been removed from other changes.*

```
9 function quickSort(uint256[] memory arr, uint left, uint right)
    ↪ public pure {
```

CVF-34. FIXED

- **Category** Suboptimal
- **Source** MathHelper.sol

Description Sorting the whole array just to find its median is waste of gas.

Recommendation Consider using the quickselect algorithm instead.

Client Comment *Code has been removed from other changes.*

```
32 quickSort(sortedArr, 0, uint(sortedArr.length - 1));
```

9 Minor Issues

CVF-35. INFO

- **Category** Procedural
- **Source** ModelerLibrary.sol

Recommendation Events are usually named via nouns.

Client Comment *Event names will not change.*

```
12 event ValidatorOptedIn(address indexed validator);
   event ValidatorOptedOut(address indexed validator, uint256
      ↪ optOutTime);
   event ModelerOptedIn(address indexed modeler);
   event ModelerOptedOut(address indexed modeler, uint256 optOutTime);
   event ValidatorRegistered(address indexed validator);
```

CVF-37. FIXED

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

Recommendation The type of the ”validatorToken” argument should be ”IERC20”.

Client Comment *I have amended the argument and internal use of safeTransferFrom.*

```
18 function registerValidator(address[] storage validatorAddresses,
      ↪ uint256 MAX_VALIDATORS, address validatorToken, DataTypes.
      ↪ ValidatorData storage validator, uint256 validatorStakeAmount)
      ↪  external {
```

CVF-38. FIXED

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

Recommendation The ”1 days” value should be a named constant.

Client Comment *Created a constant variable for this in modelerlibrary.sol.*

```
20 require(validator.isRegistered == false || (validator.optOutTime > 0
   ↪  && validator.optOutTime < block.timestamp - 1 days) , "
   ↪ Validator␣already␣registered");
```

```
56 modeler.optOutTime = block.timestamp + 1 days; //modeler must be
   ↪ online for the next 24 hours or they may be slashed
```

```
80 validator.optOutTime = block.timestamp + 1 days; //validator must be
   ↪  online for the next 24 hours or they may be slashed
```

CVF-39. FIXED

- **Category** Suboptimal
- **Source** ModelerLibrary.sol

Description This way of preventing duplicate validators is inefficient.

Recommendation Consider maintaining a mapping from validators into boolean flags telling whether a particular validator is registered.

Client Comment *Amended the code with this suggestion.*

```
30 for (uint256 i = 0; i < validatorAddresses.length; i++) {
       if (validatorAddresses[i] == _validator) {
           return;
```

CVF-40. FIXED

- **Category** Suboptimal
- **Source** ModelerLibrary.sol

Description Linear search is inefficient.

Recommendation Consider either maintaining a mapping from registered validators to their indexes or allowing the caller to provide a validator index hint.

Client Comment *Amended the code with this suggestion.*

```
40 for (uint256 i = 0; i < validatorAddresses.length; i++) {
       if (validatorAddresses[i] == _validator) {
           index = i;
           break;
       }
   }
```

CVF-41. FIXED

- **Category** Suboptimal
- **Source** ModelerLibrary.sol

Description This check is suboptimal.

Recommendation Consider refactoring the function like this: for (...) { if (...) { // remove validator from the array return; } } revert ...;

Client Comment *Amended the code with this suggestion.*

```
46 require(index < validatorAddresses.length, "Address␣not␣found");
```

CVF-42. FIXED

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

Description This should be done only if index is not length -1.

Client Comment *Amended with point 42 on this spreadsheet.*

```
47 validatorAddresses[index] = validatorAddresses[validatorAddresses.
      ↪ length - 1];
```

CVF-43. FIXED

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

Recommendation The type of the ”competitionContract” argument should be ”ICompetition”.

Client Comment *Amended the code with this suggestion.*

```
51 function optOutModeler(DataTypes.ModelerData storage modeler,
   ↪ address competitionContract) external {
```

CVF-44. FIXED

- **Category** Procedural
- **Source** IMathHelper.sol

Recommendation Consider specifying as ”^0.8.0” unless there is something special about this particular version. Also relevant for: IConsumption.sol, ICompetition.sol, ICompetitionFactory.sol, IModeler.sol, IServiceAgreement.sol, DataTypes.sol, ServiceAgreement.sol, Competition.sol, CompetitionFactory.sol, Consumption.sol, SpectralToken.sol, BeaconProxyDeployer.sol, Modeler.sol.

Client Comment *We will update to the latest version for all contracts.*

```
1 pragma solidity ^0.8.12;
```

CVF-45. FIXED

- **Category** Bad naming
- **Source** IMathHelper.sol

Description This function seems too high-level and too business-specific, for an interface named ”IMathHelper”.

Recommendation Consider moving it elsewhere or renaming the interface.

Client Comment *Renamed the contract to DataUtility.*

```
6 function generateRequestID(string calldata _input, uint256
   ↪ _blockTimestamp, address _sender) external view  returns (
   ↪ bytes32);
```

CVF-54. INFO

- **Category** Bad naming
- **Source** ICompetition.sol

Recommendation Events are usually named via nouns, such as ”Modeler”.

Client Comment *Event names will not change.*

```
7 event ModelerRegistered(address indexed modeler, string modelHash);
```

CVF-56. FIXED

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

Recommendation The type of this argument should be ”IMathHelper”.

Client Comment *Amended the code to reflect this change,*

```
11 address _mathHelper,
```

CVF-57. FIXED

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

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

Client Comment *SignUpToCompetition calls the modeler contract and that emits an event, so I believe this particular function call is handled. I have added an event all the other functions. For setModelerNetPerformanceResult, I have added an event in insertNewModeler() as this is the function that is called when there is state change.*

```
17 function signUpToCompetition(string calldata _modelHash) external;
   function setModelerNetPerformanceResultAndUpdate(address modeler,
      ↪ uint256 medianPerformanceResults) external;
   function setModelerContract(address _modelerContract) external;
20 function setConsumptionContract(address _consumptionContract)
      ↪ external;
```

CVF-59. INFO

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

Recommendation The argument type should be ”IModeler”.

Client Comment - *getModelerToMedian expects a users wallet address - isTopNModeler expects a users wallet address*
I have amended the other function setModelerContract

```
19 function setModelerContract(address _modelerContract) external;
```

```
22 function getModelerToMedian(address _modeler) external view returns
   ↪ (uint256);
```

```
25 function isTopNModeler(address _modeler) external view returns (bool
   ↪  isTopN);
```

CVF-60. FIXED

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

Recommendation The argument type should be ”IConsumption”.

Client Comment *Amended the code to reflect this change.*

```
20 function setConsumptionContract(address _consumptionContract)
   ↪ external;
```

CVF-61. INFO

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

Recommendation The return type should be ”IModeler[]”.

Client Comment *This should return an array of user addresses.*

```
21 function getTopNModelers() external view returns (address[] memory);
```

CVF-62. FIXED

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

Recommendation The return type should be ”IModeler”.

Client Comment *Amended the code to reflect this change.*

```
26 function modelerContract() external view returns (address);
```

CVF-63. FIXED

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

Recommendation The return type should be ”IServiceAgreement”.

Client Comment *Amended the code to reflect this change.*

```
27 function serviceAgreement() external view returns (address);
```

CVF-65. INFO

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

Recommendation Events are usually named via nouns, such as ”Competition” or ”Admin”.

Client Comment *Event names will not change.*

```
7 event CompetitionCreated(address indexed competition, uint256 index)
    ↪ ;
event NewAdminSet(address indexed newAdmin);
```

CVF-66. FIXED

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

Recommendation The type of the ”competition” argument should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
7 event CompetitionCreated(address indexed competition, uint256 index)
    ↪ ;
```

CVF-67. FIXED

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

Recommendation The type of the ”_competition” argument should be ”ICompetition”.

Client Comment *This function has been removed or renamed.*

```
13 function setCompetitionModelerContract(address _competition, address
    ↪  _modelerContract) external;
```

CVF-68. FIXED

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

Recommendation The type of the ”_modelerContract” argument should be ”IModeler”.

Client Comment *This function has been removed or renamed.*

```
13 function setCompetitionModelerContract(address _competition, address
    ↪  _modelerContract) external;
```

CVF-69. INFO

- **Category** Bad naming
- **Source** IModeler.sol

Recommendation Events are usually named via nouns, such as ”Validator” or ”NextRandSlot”.

Client Comment *Event names will not change.*

```
7  event ValidatorRegistered(address indexed validator);
   event NextRandSlotSet(address indexed modeler, uint256
      ↪ futureRandSlot);
   event DrandProofProvided(DataTypes.DrandProof proof);
10 event ModelerRegistered(address indexed modeler, string
      ↪ modelCommitment, uint256 stakedAmount);
   event ChallengeGiven(string ipfsChallenge, address indexed modeler,
      ↪ DataTypes.DrandProof proof);
   event ChallengeResponded(address indexed validator, string
      ↪ ipfsResponse);
   event GradedPosted(address indexed modeler, string ipfsGraded,
      ↪ uint256 performanceResult);
   event ModelerNetPerformanceResultUpdated(address indexed modeler);
   event ModelerOptedOut(address indexed modeler);
   event ModelerSlashed(address indexed modeler, uint256 amount);
```

CVF-77. FIXED

- **Category** Procedural
- **Source** DataTypes.sol

Recommendation This commented out stuff should be removed.

Client Comment *Amended the code to reflect this change.*

```
8         //uint256 minimumResponses;
```

```
21        //address[] modelersValidated;
          //modelers and their randao slots
          //mapping(address => uint256) futureRandSlots;
```

```
57        //address[] modelersRequested;
```

```
63        //InferenceType t;
```

```
65        //string ipsLink;
```

```
68  /*
        struct InferenceDispute {
70          address disputer;
            address modeler;
            uint256 disputeTime;
            string disputeData;
            bool resolved;
        }
        */
```

```
78      //enum InferenceType {INFERENCE, INFERENCE_LINK}
    /*
80      struct Inferences {
            // Desirable: monitoring number of successful requests
               ↪ catered to (for ZKML/bounty/fee sharing)
            uint256[] requestIdAssigned;
            // Mapping of request ID to inference output
            mapping(uint256 => uint256) inference;
            //InferenceType t;
        }*/
```

CVF-80. INFO

- **Category** Bad naming
- **Source** ServiceAgreement.sol

Recommendation Events are usually named via nouns, such as ”ServiceAgreement”.

Client Comment *Event names will not change.*

```
12 event ServiceAgreementCreated(DataTypes.SAParams params);
```

CVF-81. FIXED

- **Category** Suboptimal
- **Source** ServiceAgreement.sol

Recommendation The value could be rendered as ”28 days”.

Client Comment *Amended the code to reflect this change.*

```
26 require(block.timestamp > lastUpdate + 1692748800, "Service␣
   ↪ agreement␣can␣only␣be␣updated␣once␣every␣28␣days");
```

CVF-82. FIXED

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

Recommendation The minimum update period should be a named constant.

Client Comment *Amended the code to reflect this change.*

```
26 require(block.timestamp > lastUpdate + 1692748800, "Service␣
   ↪ agreement␣can␣only␣be␣updated␣once␣every␣28␣days");
```

CVF-84. FIXED

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

Recommendation The return type should be ”IERC20”.

Client Comment *Amended the code to reflect this change.*

```
59 function stakedToken() external view returns (address) {
```

CVF-91. INFO

- **Category** Bad naming
- **Source** Competition.sol

Recommendation Events are usually named via nouns, such as ”Modeler” or ”TopNModelers”.

Client Comment *Event names will not change.*

```
36 event ModelerRegistered(address indexed modeler, string modelHash);
   event TopNModelersUpdated(address[] topNModelers);
   event ServiceAgreementSigned(address indexed modeler);
```

CVF-94. FIXED

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

Description These functions should emit some events.

Client Comment *Amended the code to reflect this change.*

```
75  function setModelerNetPerformanceResultAndUpdate(address modeler,
       ↪ uint256 medianPerformanceResults) external onlyModelerContract
       ↪  {
```

```
119 function replaceOptOutTopNModeler(address _oldModeler, address
       ↪ _newModeler, uint256 _medianPerformanceResults) external
       ↪ onlyModelerContract {
```

```
132 function setModelerContract(address _modelerContract) external
       ↪ onlyAdmin {
```

```
137 function setConsumptionContract(address _consumptionContract)
       ↪ external onlyAdmin {
```

```
142 function setIPFSCompetitionDescriptionHash(string calldata
       ↪ _ipfsCompetition) external onlyAdmin {
```

```
147 function setIPFSTrainingDataSet(string calldata _ipfsTrainingDataset
       ↪ ) external onlyAdmin {
```

```
177 function emergencyOptOutValidator(address _validator) external
       ↪ onlyAdmin {
```

```
181 function setAdmin(address _newAdmin) external onlyAdmin {
```

CVF-95. FIXED

- **Category** Suboptimal
- **Source** Competition.sol

Description The expression ”topNModelers.length” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
79 if (medianPerformanceResults > modelerToMedian[topNModelers[
   ↪ topNModelers.length - 1]]) {
80     address lowestModeler = topNModelers[topNModelers.length - 1];
       topNModelers[topNModelers.length - 1] = modeler;
```

```
84     quickSort(0, topNModelers.length - 1);
```

CVF-96. FIXED

- **Category** Suboptimal
- **Source** Competition.sol

Description The expression ”topNModelers[topNModelers.length -1]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
79 if (medianPerformanceResults > modelerToMedian[topNModelers[
   ↪ topNModelers.length - 1]]) {
80     address lowestModeler = topNModelers[topNModelers.length - 1];
```

CVF-101. FIXED

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

Recommendation The type for this variable should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
12 address private competitionLogic;
```

CVF-102. FIXED

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

Recommendation The type for this variable should be ”IModeler”.

Client Comment *Amended the code to reflect this change.*

```
13 address private modelerLogic;
```

CVF-103. FIXED

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

Recommendation The type for this variable should be ”ICompetition[]”.

Client Comment *Amended the code to reflect this change.*

```
19 address[] public allCompetitions;
```

CVF-104. FIXED

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

Recommendation The type for this variable should be ”IModeler[]”.

Client Comment *Amended the code to reflect this change.*

```
20 address[] public allModelerContracts;
```

CVF-105. INFO

- **Category** Bad naming
- **Source** CompetitionFactory.sol

Recommendation Events are usually named via nouns, such as ”Competition”, ”CompetitionUpgrade”, or ”ModelerContract”.

Client Comment *Event names will not change.*

```
22 event CompetitionCreated(address indexed competition, uint256 index)
   ↪ ;
   event CompetitionUpgraded(address indexed upgrader, address
   ↪ competition);
   event ModelerContractCreated(address indexed modelerContract,
   ↪ uint256 index);
   event ModelerContractUpgraded(address indexed upgrader, address
   ↪ modelerContract);
   event NewAdminSet(address indexed newAdmin);
```

CVF-106. FIXED

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

Recommendation The type for the ”competition” parameter should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
22 event CompetitionCreated(address indexed competition, uint256 index)
   ↪ ;
   event CompetitionUpgraded(address indexed upgrader, address
   ↪ competition);
```

CVF-107. FIXED

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

Recommendation The type for the ”modelerContract” parameter should be ”IModeler”.

Client Comment *Amended the code to reflect this change.*

```
24 event ModelerContractCreated(address indexed modelerContract,
      ↪ uint256 index);
   event ModelerContractUpgraded(address indexed upgrader, address
      ↪ modelerContract);
```

CVF-108. FIXED

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

Recommendation The type for this variable should be ”IMathHelper”.

Client Comment *Amended the code to reflect this change.*

```
29 address public mathHelper;
```

CVF-109. FIXED

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

Recommendation The argument type should be ”IMathHelper”.

Client Comment *Amended the code to reflect this change.*

```
33 constructor(address _mathHelper) {
```

CVF-110. FIXED

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

Recommendation The arguments types should be ”ICompetition” and ”IModeler” respectively.

Client Comment *Amended the code to reflect this change.*

```
38 function initialize(address _competitionLogic, address _modelerLogic
   ↪ ) external initializer {
```

CVF-111. FIXED

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

Recommendation The return type should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
55 ) external returns (address) {
```

CVF-112. FIXED

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

Recommendation The type for this argument should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
74 address _competitionContract,
```

CVF-113. FIXED

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

Recommendation The type for this argument should be ”IERC20”.

Client Comment *Amended the code to reflect this change.*

```
75 address _validatorToken,
```

CVF-114. FIXED

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

Recommendation The return type should be ”IModeler”.

Client Comment *Amended the code to reflect this change.*

```
78 ) external returns (address) {
```

CVF-115. FIXED

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

Recommendation The argument should be ”ICompetition”.

Client Comment *Amended the code to reflect this change.*

```
95 function upgradeCompetition(address newCompetition) external {
```

CVF-116. FIXED

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

Recommendation The argument type should be ”IModeler”.

Client Comment *Amended the code to reflect this change.*

```
104 function upgradeModelerContract(address newModelerContract) external
    ↪ {
```

CVF-117. FIXED

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

Recommendation The return type should be ”ICompetition[]”.

Client Comment *Amended the code to reflect this change.*

```
117 function getAllCompetitions() public view returns (address[] memory)
      ↪  {
```

CVF-118. FIXED

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

Recommendation The return type should be ”IModeler[]”.

Client Comment *Amended the code to reflect this change.*

```
125 function getAllModelersContracts() public view returns (address[]
      ↪ memory) {
```

CVF-140. FIXED

- **Category** Suboptimal
- **Source** MathHelper.sol

Description Type conversions are redundant, as ”i” and ”j” are already ”uint256”.

Client Comment *Amended the code to reflect this change.*

```
15 while (arr[uint256(i)] < pivot) i++;
   while (pivot < arr[uint256(j)]) j--;
```

```
18     (arr[uint256(i)], arr[uint256(j)]) = (arr[uint256(j)], arr[
         ↪ uint256(i)]);
```

CVF-141. FIXED

- **Category** Procedural
- **Source** Modeler.sol

Description Variables are usually not named IN_UPPER_CASE.

Recommendation Consider renaming inCamelCase.

Client Comment *Amended the code to reflect this change.*

```
13 uint256 MAX_VALIDATORS;
```

CVF-142. FIXED

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

Recommendation The type of this variable should be ”IERC20”.

Client Comment *Amended the code to reflect this change.*

```
14 address validatorToken;
```

CVF-144. INFO

- **Category** Suboptimal
- **Source** Modeler.sol

Recommendation It would be more efficient to merge these mappings into a single mapping whose keys are modelers and values are structs encapsulating the values of the original mappings.

Client Comment *Acknowledged, we would like to keep them separate for readability.*

```
18 mapping(address => DataTypes.ModelerData) public modelers;
```

```
20 mapping(address => DataTypes.ModelerChallenge) public
   ↪ modelerChallenges;
```

CVF-145. FIXED

- **Category** Documentation
- **Source** Modeler.sol

Description The semantics of the keys for this mapping is unclear.

Recommendation Consider documenting.

Client Comment *Added a comment above this mapping.*

```
21 mapping(address => DataTypes.ModelerChallenge) public ZKMLChallenges
   ↪ ;
```

CVF-148. INFO

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

Recommendation The type for this variable should be ”ICompetition”.

Client Comment *Acknowledged, our contract has reached beyond bytecode limit and this check is not necessary for core functionality but increases gas cost*

```
30 address competitionContract;
```

CVF-149. INFO

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

Recommendation The type for this variable should be ”IConsumption”.

Client Comment *Acknowledged, our contract has reached beyond bytecode limit and this check is not necessary for core functionality but increases gas cost*

```
31 address consumptionContract;
```

CVF-151. INFO

- **Category** Bad naming
- **Source** Modeler.sol

Recommendation Events are usually named via nouns, such as ”Challenge”, ”ModelerResponse” etc.

Client Comment *Acknowledged, we feel that the events are understandable*

```
35 event ChallengeGiven(address indexed validator, address indexed
      ↪ modeler, string ipfsChallenge);
   event ModelerResponds(address indexed validator, address indexed
      ↪ modeler, string ipfsResponse);
   event ModelerGraded(address indexed validator, address indexed
      ↪ modeler, string ipfsGraded);
   event ModelUpdated(address indexed modeler, string modelCommitment);
   event SetRandSlot(address indexed validator, address indexed modeler
      ↪ , uint256 randSlot);
40 event ZKMLChallengeGiven(address indexed validator, address indexed
      ↪ modeler, string ipfsChallenge);
```

```
42 event DrandSubmitted(address indexed validator, uint256 round,
      ↪ bytes32 randomness, bytes signature, bytes previous_signature)
      ↪ ;
   event ModelerKicked(address indexed validator, address indexed
      ↪ modeler);
```

CVF-152. FIXED

- **Category** Suboptimal
- **Source** Modeler.sol

Description This constructor is redundant.

Recommendation Consider removing it.

Client Comment *Amended the code to reflect this change.*

```
60 constructor(){}
```

CVF-153. FIXED

- **Category** Procedural
- **Source** Modeler.sol

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

Client Comment *Code has been removed as per 152.*

```
60 constructor(){}
```

CVF-154. INFO

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

Recommendation The type for this argument should be ”ICompetition”.

Client Comment *Acknowledged, our contract has reached beyond bytecode limit and this check is not necessary for core functionality but increases gas cost*

```
63 address _competitionContract,
```

CVF-155. FIXED

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

Recommendation The type for this argument should be ”IERC20”.

Client Comment *Amended the code to reflect this change.*

```
64 address _validatorToken,
```

CVF-157. FIXED

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

Description This function should emit some event.

Client Comment *Amended the code to reflect this change.*

```
86 function setIPFSDataSet(string calldata _ipfsTestingDataset)
      ↪ external onlyValidator {
```

CVF-159. INFO

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelers[msg.sender]” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *As we are over the bytecode limit, we will leave this inefficiency in the code to save deployment gas.*

```
104 require(modelers[msg.sender].modelerSubmissionBlock != 0, "Modeler␣
      ↪ not␣registered");
uint256 stakedAmount = modelers[msg.sender].currentStake;
modelers[msg.sender] = DataTypes.ModelerData(_modelCommitment, block
      ↪ .number, 0, stakedAmount, ~uint256(0));
```

```
109 modelers[msg.sender].medianPerformanceResults = 0;
```

CVF-160. INFO

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelerChallenges[msg.sender]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
107 modelerChallenges[msg.sender].ipfsGraded = "0";
modelerChallenges[msg.sender].performanceResult = 0;
```

CVF-161. INFO

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelerChallenges[msg.sender]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *As we are over the bytecode limit, we will leave this inefficiency in the code to save deployment gas.*

```
123 require(bytes(modelerChallenges[msg.sender].ipfsChallenge).length >
       ↪ 0, "Challenge␣not␣found");
    modelerChallenges[msg.sender].ipfsResponse = _ipfsResponse;
```

CVF-162. INFO

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelerChallenges[modeler]” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *As we are over the bytecode limit, we will leave this inefficiency in the code to save deployment gas.*

```
129 require(bytes(modelerChallenges[modeler].ipfsChallenge).length > 0,
       ↪ "Modeler␣not␣been␣challenged");
130 require(bytes(modelerChallenges[modeler].ipfsResponse).length > 0, "
       ↪ Modeler␣not␣responded␣to␣challenge");
    modelerChallenges[modeler].ipfsGraded = _ipfsGraded;
    modelerChallenges[modeler].performanceResult = _performanceResult;
    modelerChallenges[modeler].ipfsGradeDetailLink =
       ↪ _ipfsGradeDetailLink;
```

CVF-163. FIXED

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelers[modeler]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
140 require(modelers[modeler].optOutTime > block.timestamp, "Already␣
       ↪ opted␣out");
    require(modelers[modeler].currentStake >= IServiceAgreement(
       ↪ ICompetition(competitionContract).serviceAgreement()).
       ↪ stakedAmount(), "Modeler␣does␣not␣have␣enough␣stake␣to␣
       ↪ participate");
```

CVF-164. FIXED

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”ZKMLChallenge[msg.sender]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
155 require(bytes(ZKMLChallenges[msg.sender].ipfsChallenge).length > 0,
       ↪ "Challenge␣not␣found");
    ZKMLChallenges[msg.sender].ipfsResponse = _ipfsResponse;
```

CVF-166. FIXED

- **Category** Suboptimal
- **Source** Modeler.sol

Description The expression ”modelers[modeler]” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Amended the code to reflect this change.*

```
162 require(modelers[modeler].optOutTime > block.timestamp, "New␣modeler
       ↪ ␣opted␣out");
    require(modelers[modeler].currentStake >= IServiceAgreement(
       ↪ ICompetition(competitionContract).serviceAgreement()).
       ↪ stakedAmount(), "New␣modeler␣does␣not␣have␣enough␣stake␣to␣
       ↪ participate");
```

CVF-167. INFO

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

Recommendation The argument type should be ”IConsumption”.

Client Comment *Acknowledged, our contract has reached beyond bytecode limit and this check is not necessary for core functionality but increases gas cost*

```
209 function setConsumptionContract(address _consumptionContract)
       ↪ external onlyCompetition whenNotPaused {
```

CVF-168. FIXED

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

Recommendation The parameter type should be ”UpgradeableBeacon”.

Client Comment *Amended the code to reflect this change.*

```
13 event BeaconDeployed(address _beacon);
```

CVF-172. FIXED

- **Category** Suboptimal
- **Source** BeaconProxyDeployer.sol

Description These two functions are very similar.

Recommendation Consider extracting common parts into utility functions.

Client Comment *Amended the code to reflect this change.*

```
44 function deployCompetitionBeaconProxy(address logic, bytes memory
   ↪ payload) internal returns (address) {
```

```
62 function deployModelerBeaconProxy(address logic, bytes memory
   ↪ payload) internal returns (address) {
```

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