A CONSENSYS DILIGENCE AUDIT REPORT

iExec PoCo

1 Executive Summary

Date	March 2020
Lead Auditor	Gonçalo Sá
Co-auditors	Shayan Eskandari

This report presents the results of our engagement with **iExec** to review their **PoCo (Proof of Contribution)** protocol.

The review was conducted over the course of two weeks, from **March 30, 2020** to **April 10, 2020** by Gonçalo Sá and Shayan Eskandari. A total of **15** person-days were spent.

During the first week, we focused our efforts on understanding the intention of the design (which is mostly provided through communication with the client and the resources provided in the README of the main repository under review, `poco-dev`), and defining the key risk factors and potential vulnerabilities requiring further investigation. We also initiated an isolated code review of the `iexec-solidity` repository, still not considering interactions with the `poco-dev` codebase.

During the second week we initiated the code review efforts for both repositories under review. Focusing on interactions between the two repositories and a standalone review of the ERC1538 delegates present in the `poco-dev` repository.

2 Scope

Our review focused on two repositories:

- https://github.com/iExecBlockchainComputing/poco-dev.git @ a4dfe7891ac60489809cdd4d9c491c8f2e107a82
- https://github.com/iExecBlockchainComputing/iexec-solidity.git @ a4dfe7891ac60489809cdd4d9c491c8f2e107a82

The list of files in scope can be found in the Appendix.

They represent the big majority of files that comprise the iExec system (the only exception being the RLC token dependencies that remain unchanged throughout multiple versions for the PoCo system). Note that many of the check and effects of the iExec platform are done off-chain and not in the scope of this audit.

The allotted time for for the audit (three person-weeks over the span of two weeks time) was deemed insufficient from the start to do a full comprehensive review of the whole system. And, even reducing the amount of visual collateral being provided as part of the report, some compromises had to be made on the completeness of the audit.

As such, this audit is mostly **focused on the correctness of the code** in individual modules and less so on the adhesion to the specification of the business logic of the Proof of Contribution system. In addition, there are some mathematical models that have been modified to fit into solidity variables, such as the implementation of **trust** variable (e.g. floating point to integer, see Trust in the PoCo), the mathematics behind the conversion falls outside the scope of this audit and only the correctness of client's implementation was reviewed.

2.1 Documentations

The following documentations were provided to the audit team:

- PoCo Series #1 — About Trust and Agents Incentives
- PoCo Series #2 — On the use of staking to prevent attacks

- PoCo Series #3 — Protocol update
- PoCo Series #4 — Enclaves and Trusted Executions
- PoCo Series #5 — Open decentralized brokering on the iExec platform
- Proof of Contribution - docs.iex.ec
- iExec platform documentation: Trust in the PoCo

2.2 Objectives

Through discussion with the **iExec** team, we identified the following priorities for our review

1. Ensure code correctness in each individual module in the system.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.
3. Make sure each module is implemented consistently with the intended functionality and without unintended edge cases.

3 System Overview

The iExec platform uses blockchain technology to create a marketplace where people can rent computing power to run Applications provided by App developers and/or use Datasets provided Dataset providers.

The iExec platform requires two entities in order to work, and PoCo acts as a link between those two entities:

- A marketplace where agents propose their resources and where deals are made using the RLC token.
- A distributed computing infrastructure based on the middleware XtremWeb-HEP.

3.1 PoCo Delegate

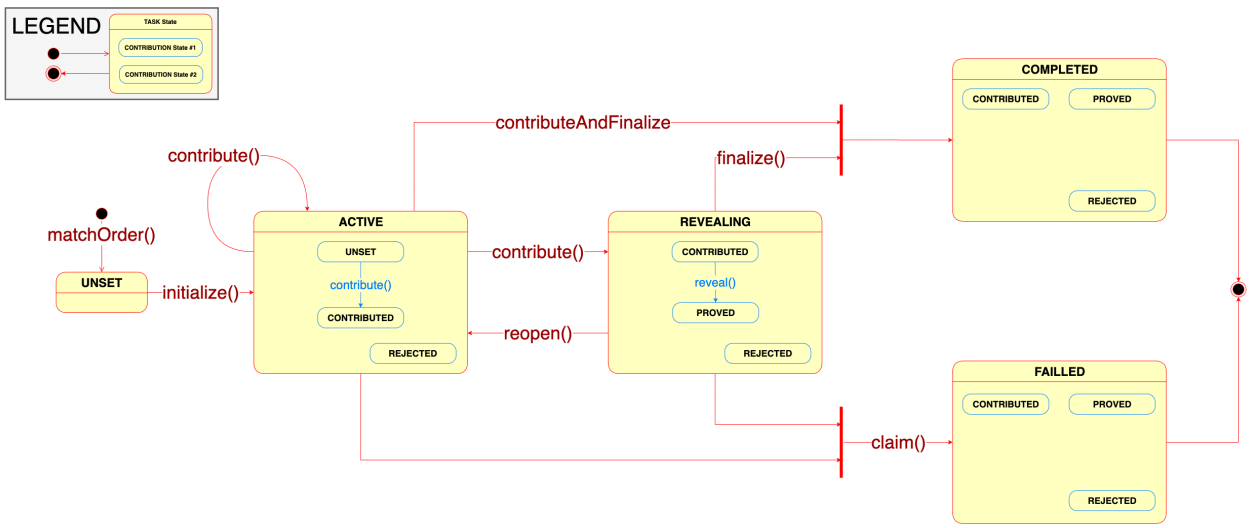
The core part of the PoCo system is the new `PoCoDelegate` smart contract. It replaces what used to be a combination of two smart contracts: the `IexecClerk` and the `IexecHub`.

The PoCo delegate (which, as the name indicates, is a delegate for the ERC1538 proxy acting as the entry for the system) implements almost all of the logic that rules over the success or failure of deals and, more specifically, tasks in the iExec system.

`IexecPoCoDelegate` is, undoubtedly, the most important smart contract of the PoCo architecture and it's inception as a single smart contract is new to version 5 of the system.

The PoCo delegate handles both the token escrow and validation of the submitted computation results and handles the permissions (most of them through the checking of signatures from the relevant parties) of all the actors.

A PoCo delegate state diagram was generated to map out the state machines for both `Tasks` and `Contributions`, two important data structures for iExec's business logic, and guide the audit team through the review of the code.



The state machine, although complex, is clearly implemented in the code, with clear requirements enclosing the relevant functions.

3.2 Actors

In this platform, there are 4 main types of agents:

- **Application providers** provide applications running on the Ethereum/iExec platform and receive payments in RLC.

    - **Dataset providers** provide valuable datasets in a secure paradigm to protect their ownership.
- **Users** want to run applications and are therefore buying computing power to execute them.
- **Workers** execute applications required by the user and are therefore selling computing power. They receive payments in RLC for the computation power they provide. Workers can be pooled together in *worker pools*, and will be led by scheduler for work distribution.
- **Schedulers** organize workers into working pools and manage the execution of tasks: handling work distribution, assigning tasks to workers, transferring data, and handling failures. They do not do the actual computation, however they receive a fee for managing the infrastructure. Scheduler is also responsible for *random* worker selection.

iExec Hub & Market place: Smart contract without any privilege access to act as an *escrow* for the different agents' stake and provide transparency in the iExec ecosystem. Also *workers' reputation* is stored in this contract to enable workers to switch schedulers at will.

More on the permissions and ability of each actor can be found in Security Specification section.

4 Additional Spot Check of Uniswap's Token Swap Delegate

An additional 1-day spot check was performed on the 21st of September of 2020 to validate a small addition to the PoCo codebase encompassed in the following PR: iExecBlockchainComputing/PoCo#45

Aside from helper functions added to existing files (e.g., https://github.com/iExecBlockchainComputing/PoCo/pull/45/files#diff-1e73aff6367b2514e7d2d69b9dc56a91R83-R90), the significant change in the PoCo logic is the addition of the `IexecEscrowTokenSwapDelegate` contract.

The intent of this new delegate contract is to enable atomic Uniswap swaps when depositing ETH to the PoCo system or, more importantly, when matching orders in the PoCo system.

The external functions present in:

https://github.com/iExecBlockchainComputing/PoCo/blob/c9b9bbb39129a0596b3e78b891d7f4be11c892b7/contracts/modules/delegates/IexecEscrowTokenSwapDelegate.sol#L96-L105

Are just wrapper functions over the Uniswap v2 router methods for paths specifically including RLC, iExec's native token, and, therefore, the attack surface they open up is limited.

The most crucial logic addition is in the internal functions handling the token swaps and an extra external function serving as another wrapper to match orders in the PoCo system with ETH while swapping it atomically on Uniswap.

In the internal functions, the Checks-Effects-Interactions pattern is correctly employed. However, given necessity, in the `matchOrdersWithEth` function, the call to the internal `_request` function opens up the possibility of reentrancy by sending back excess ETH to the sender.

The `m_consumed[]` state variable (a mapping) is accessed before the external call inside `_request`, and, therefore, it is not guaranteed that its state can be guaranteed before the call to the `matchOrders` function in `IexecPocoDelegate`.

However, after careful analysis, we can see that `m_consumed[]` is monotonic, which in turn means that the volume of each order will only strictly decrease. Additionally, we can also verify that, in the event the available order volume is not enough, the call will revert.

Tying this together, we conclude that the reliance on the consistent state of `m_consumed[]` before and after the reentrancy is not problematic because it could only result in lost funds for an ill-intended actor and, as a result, there is no incentive to change `m_consumed[]`'s state with the reentrancy.

One small detail that might be worth to mention in user-facing communications from iExec is that the caller of the `matchOrdersWithEth` function in `IexecEscrowTokenSwapDelegate` will always be donating the proceeds of the swap to the order requester and not to himself. However, the audit team is not disagreeing with the way the module is engineered since if the

proceeds went to `msg.sender` a class of vulnerabilities would have been surfaced.

5 Recommendations

5.1 Avoid memory manipulation routines in assembly

Even though the gas optimizations stemming from direct memory manipulation routines in assembly is commendable (these are mostly present in hashing-related functions in the `IexecLibCore_v5` library), the average saved gas per function is close to `600 gas` only. This means that, in average, a few thousand gas per user call will be saved at the expense of a big reduction in readability and auditability.

The audit team suggests that vanilla Solidity patterns are used in place of the more custom assembly blocks present in the code.

Update: iExec team agreed with this suggestion and implemented a fix in PoCo-dev/pull/70/.

5.2 Avoid repeated code throughout the codebase

There are several instances of repeated contracts and code snippets throughout the two repositories under review. In some cases even differing slightly in the actual implementations. An effort should be made to reduce these duplicated instances to a minimum and, when possible, eliminate duplication at all.

Update: iExec team agreed with this suggestion and implemented a fix in a74542102a1c4969eca8fef0f947581f4f834a4c.

5.3 Consider replacing the ERC1538 standard

Consider using a more simplistic and auditable version of delegation than implementing the full ERC1538 standard. The two scenarios where delegation might be needed are covered below.

For size-constraint purposes, a simple fallback delegating to a following contract (this can, obviously, be a chain of multiple contracts in case the original contract is too big).

For purposes of gas optimization, external calls might still result in cheaper execution costs in the long run because of the additional cost of executing the pre-delegation piece of code in the proxy.

For modularity, the same architectural structure can be achieved with normal external calls and possibly a centralized registry that allows updates.

Update from the iExec team: The feature has been planned for almost 1 year, including communication about the advantages in terms of modularity and "future-proofness". We would only consider removing the ERC1538 implementation if there was something fundamentally broken about it.

5.4 Simplify the inheritance and modularity of the system

Consider using less inheritance in similar classes for more audibility of the code. This is for overall the coding style of iExec code base. As an example discussed with the developer team, registries can be all combined together and use types for each registers.

The current implementation has 3 main registries (and corresponding entities), **apps**, **dataset**, and **workerpools**. They share most of their logic in another file `Registry.sol`. All these registries can be combined in one registry, and by adding a type Enum (or other methods) they can be differentiated.

Update from the iExec team: We need the 3 registries to be different contracts in order for the 3 classes of assets to be independent ERC721 flavors. We would like to avoid any possible confusion between apps, datasets, and workerpools. And having all 3 be the same ERC721 family would create confusion.

5.5 Correct spelling mistakes present in variable names

Even though spelling mistakes are generally harmless when writing code, they can be harmful if not made consistently. There are two instances of spelling mistakes that are used in the PoCo codebase present in the codebase inconsistently.

On the task status Enum the value *FAILLED* is spelled wrong but in the function in `PoCoDelegate` that makes sets this state is actually correctly named `failedWork()`. We recommend changing all instances of the Enum value to **FAILED**.

The other pervasive instance of a spelling mistake happens on the word *consensus* throughout the codebase. In this case, the inconsistency is only reflected in the difference from comments to the actual variable names. We recommend changing all the instances of *concensus* to **consensus** to prevent possible future errors.

Update: iExec team agreed with this suggestion and implemented a fix in 7bcbb54c8696664607a0135d02be5365abc584e2 and a7fc84f2e72e5f4acdc147601d51234fb409907f.

5.6 Review the Code Quality recommendations in Appendix 1

Other comments related to readability and best practices are listed in Appendix 1

6 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation. The purpose of this section is to identify specific security properties that were validated by the audit team.

6.1 Trust Model

The relevant actors are listed below with their respective abilities:

System Deployer (iExec)

- Initially deploys and configures the iExec system, such as setting the address for the `baseToken`, all registries and iExec hub
- Upgrade and change the main contracts (registries):
  - App Registry

    - Dataset Registry
    - Worker Pool Registry
- Escrow Modifications
    - Recover funds and add to owner balance `recover()`
- Set callback gas limit `m_callbackgas`

Scheduler

- Manages Requests:
    - Reopen closed request `reopen()`
    - Finalize requests and contributions, which results in reward distribution to workers
- Manage Worker Pool Operation
- Create Worker pools, Set and Change policy of the worker pool, such as Stake ratio and Reward Ratio policies
- Sign PoolOrder for the work they are matching with

Worker (Computation Power Provider)

- Contribute work to tasks `contribute()`
- Reveal the contributed work `reveal()`

App Developer

- Create app `createApp()`
- Manage their submitted app `manageAppOrder()`
- Sign AppOrder for their app

Dataset Provider

- Create dataset `createDataset()`
- Manage their submitted dataset `manageDatasetOrder()`
- Sign DataOrder for their Datasets

Platform User (Computation Power Buyer)

- Request a task to be perform and stakes tokens for the requested computation
- Manage their submitted request `manageRequestOrder()`

- Sign the requestOrder

Note that App and Dataset signatures are assumed to be available publicly for users to use in their request orders. Workerpool and users signatures are gathered off-chain during the order request and bundled together with the App and Dataset signature to be sent to iExec hub (e.g. `matchOrder()`).

6.2 Funds

- All *actors* can deposit RLC on the iExec Hub.
- Funds deposited on the *iExec Hub* can be locked when staking. iExec Hub also holds all deposited rewards.
    - Funds that are not actively staked (locked) can be withdrawn at any time.
- *Worker*'s stake in WorkerPool: This stake cannot be seized by anyone, and the worker can unlock it at anytime (by unsubscribing). Even If the worker is evicted by the scheduler (presumably because of a bad behavior) its stake will be unlocked.

It should be noted that the contracts that are named `Native` (such as `IexecEscrowNativeDelegate.sol`) are assumed to be deployed on iExec side chain and are not considered for mainnet deployment.

6.3 Important Security Properties

The following is a non-exhaustive list of security properties that were verified in this audit.

`iexec-solidity` Repository

- All the meant-to-be-internal, state-changing functions are correctly marked internal.
- All the external accessing functions accessing internal functions that can change the proxy's state (which functions it delegates to) are correctly permeated by `Ownable` -inherited modifiers.
- Delegates in the repository with state-changing methods (only the `Update` delegate) have correctly permeated functions with `onlyOwner` .
- The inheritance tree and delegation system of the ERC1538 architecture of the contract system are correctly implemented and do

not create problems with shadowed elements or unimplemented methods.
- No unsigned integers in LibMap2 methods handling array indexes can underflow.
- No unsigned integers in LibSet methods handling arrays indexes can underflow.
- The compact signature recovery (EIP 2098) is correctly implemented (as per Nick Johnson's referral implementation).

`poco-dev` Repository

- The PoCo delegate state machine is implemented according to the intents stated in the documentation.
    - *Note*: The documentation refers only to previous versions' architecture with a *Clerk* and *Hub* instead of a `PocoDelegate`. The new specification that was validated is an extrapolation of the audit team.
- The signature checking methods are correctly implemented.
- No malicious actors can withdraw tokens from other agents' escrows.
- PoCo has it's own implementation of ERC20, and it conforms with the ERC20 specification.
- PoCo delegate is inherently trusted, `owner` can upgrade the underlying contracts.
- Three registries exist that implement App, Dataset, and Workerpool. Note that they must be initialized to set proper values and only owner can change their policies.
- Management functionality for Requests, Apps, Datasets, and Workerpool scheduler are implemented as intended, with only the initial submitter being able to post the pre-signature or changing the task details.
- Structs meant for yet-to-be-implemented features are not accessible by any method in the current system.
- No problem arises from some of the *External* accessing functions being marked as *Public* (e.g., to prevent stack too deep compiler error).
- No unintended deadlock conditions arise in any part of the system

from the use of `ExtendedSafeMath` methods.
- Incentives are correctly implemented for all of the actors in the PoCo system.

7 Issues

Each issue has an assigned severity:

- **Minor** issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- **Medium** issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- **Major** issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- **Critical** issues are directly exploitable security vulnerabilities that need to be fixed.

7.1 Permissionless nature of proxy factory might cause confusion when parsing events **Acknowledged**

Resolution
Update from the iExec team: The iExec offchain platform does not listen to GenericFactory. This factory is intended to be public and available to anyone and is just a tool used for deployment.

Description

The permissionless nature of the factory (the `GenericFactory` contract)

meant to deploy the `ERC1538Proxy` and the instances of its several delegates might create confusion when parsing events.

Since there is no access control being enforced through the use of modifiers on said factory, any account can use its deployment public methods to deploy a contract. This means that the supporting off-chain infrastructure making use of the fired events to look for deployed instances of either the iExec proxies or its delegates might get hindered by an ill-intended actor that abuses its functions.

Recommendation

Use a modifier enforcing some sort of access control (easily done through the inherited `Ownable` contract) to make sure only iExec can deploy from the factory and, therefore, increase the readability of logged events.

This becomes more important as time goes by and updates to the architecture are performed or any past analysis needs to be done on deployed modules.

7.2 System deployer is fully trusted in this version of the PoCo system Medium Acknowledged

Resolution
Update from the iExec team: After deployment, ownership is planned to be transferred to a multisig. This is just the first step towards a more decentralised governance on the protocol. We will consider adding an intermediary contract that enforces the lock period. This would however, prevent us from any kind of “emergency” update. The long term goal is it involve the community in the process, using a DAO or a similar solution.

Description

The introduction of ERC1538-compliant proxies to construct the PoCo system has many benefits. It heightens modularity, reduces the number of external calls between the system's components and allows for easy expansion of the system's capabilities without disruption of the service or need for off-chain infrastructure upgrade. However, the last enumerated benefit is in fact a double-edged sword.

Even though ERC1538 enables easy upgradeability it also completely strips the PoCo system of all of its prior trustless nature. In this version the iExec development team should be entirely trusted by **every** actor in the system not to change the deployed on-chain delegates for new ones.

Also the deployer, `owner`, has permission to change some of the system variables, such as `m_callbackgas` for Oracle callback gas limit. This indirectly can lock the system, for example it could result in `IexecPocoDelegate.executeCallback()` reverting which prevents the finalization of corresponding task.

Recommendation

The best, easiest solution for the trust issue would be to immediately revoke ownership of the proxy right after deployment. This way the modular deployment would still be possible but no power to change the deployed on-chain code would exist.

A second best solution would be to force a timespan period before any change to the proxy methods (and its delegates) is made effective. This way any actor in the system can still monitor for possible changes and "leave" the system before they are implemented.

In this last option the "lock" period should, obviously, be greater than the amount of time it takes to verify a `Task` of the bigger category but it is advisable to decide on it by anthropomorphic rules and use a longer, "human-friendly" time lock of, for example, 72 hours.

7.3 `importScore()` in `IexecMaintenanceDelegate` can be used to wrongfully reset worker scores Medium

Acknowledged

Resolution

Update from the iExec team:

In order to perform this attack, one would first have to gain reputation on the new version, and lose it. They would then be able to restore its score from the old version.

We feel the risk is acceptable for a few reasons:

- It can only be done once per worker
- Considering the score dynamics discussed in the "Trust in the PoCo" document, it is more interesting for a worker to import its reputation in the beginning rather then creating a new one, since bad contributions only remove part of the reputation
- Only a handful of workers have reputation in the old system (180), and their score is low (average 7, max 22)

We might force the import all 180 workers with reputation >0. A script to identify the relevant addresses is already available.

Description

The import of worker scores from the previous PoCo system deployed on chain is made to be asynchronous. And, even though the pull pattern usually makes a system much more resilient, in this case, it opens up the possibility for an attack that undermines the trust-based game-theoretical balance the PoCo system relies on. As can be seen in the following function:

code/poco-dev/contracts/modules/delegates/IexecMaintenanceDelegate.sol:L51-L57

```
function importScore(address _worker)
external override
{
        require(!m_v3_scoreImported[_worker], "score-already-imported");
        m_workerScores[_worker] = m_workerScores[_worker].max(m_v3_iexecH
        m_v3_scoreImported[_worker] = true;
}
```

A motivated attacker could attack the system providing bogus results for computation tasks therefore reducing his own reputation (mirrored by the low worker score that would follow).

After the fact, the attacker could reset its score to the previous high value attained in the previously deployed PoCo system (`v3`) and undo all the wrongdoings he had done at no reputational cost.

Recommendation

Check that each worker interacting with the PoCo system has already imported his score. Otherwise import it synchronously with a call at the time of their first interaction.

7.4 Outdated documentation Medium Acknowledged

Resolution
Update from the iExec team: `Work in progress.`

Description

There are many changes within the system from the initial version that are not reflected in the documentation.

It is necessary to have updated documentation for the time of the audit, as the specification dictates the correct behaviour of the code base.

Examples

Entities such as `iExecClerk` are the main point of entry in the

documentation, however they have been replaced by proxy implementation in the code base (V5).

Recommendation

Up date documentation to reflect the recent changes and design in the code base.

7.5 Domain separator in `iExecMaintenanceDelegate` has a wrong version field Medium Acknowledged

Resolution
Issue was fixed in iExecBlockchainComputing/PoCo-dev@ `ebee370`

Description

The domain separator used to comply with the EIP712 standard in `iExecMaintenanceDelegate` has a wrong version field.

code/poco-dev/contracts/modules/delegates/IexecMaintenanceDelegate.sol:L77-L86

```
function _domain()
internal view returns (IexecLibOrders_v5.EIP712Domain memory)
{
        return IexecLibOrders_v5.EIP712Domain({
                name:              "iExecODB"
        , version:           "3.0-alpha"
        , chainId:           _chainId()
        , verifyingContract: address(this)
        });
}
```

In the above snippet we can see the code is still using the version field from an old version of the PoCo protocol, `"3.0-alpha"` .

Recommendation

Change the version field to: `"5.0-alpha"`

7.6 Limit the length of `task.contributors` to prevent reaching gasBlockLimit Minor Acknowledged

Resolution

Update from the iExec team:

Any hardcoded lock would be a restriction in the future if thee block size increases. In addition to that, workers are strongly incentivised to not contribute if it would result in a deadlocked task. Schedulers are incentivised to not authorise too many workers to contribute (they also lose stake if a task get deadlocked). So the development team has assessed the risk as low.

In the unlikely event the described flaw still happens, the task will get in a deadlocked state, until at some point the block size limit is increased and a claim becomes possible. Because in a world where block size increases are possible, deadlocks are not eternal.

Description

It is recommended to limit the length of arrays that the contract iterates through to prevent system halts. `task.contributors` is used within iExec contract in many functions, and main functions such as `claim()`, `reOpen()`, and most importantly `contribute()` (through calling `checkConsensus()`) iterate through this list.

Given that contributions are not free and they could only block the task they are contributing to, this is a low impact issue.

Recommendation

The fix is trivial to implement and only requires to limit the number of items in `task.contributors` to the maximum imagined for the system (based on

client communication this number could be 20, although further testing should be done to make sure with this number does not reach the blockGasLimit, possibly with future changes in the opcode pricing).

7.7 The `updateContract()` method in `ERC1538UpdateDelegate` is incorrectly implemented

Minor

Resolution
Issue was fixed in iExecBlockchainComputing/iexec-solidity@ `e6be083`

Description

The `updateContract()` method in `ERC1538UpdateDelegate` does not behave as intended for some specific streams of bytes (meant to be parsed as function signatures).

The mentioned function takes as input, among other things, a `string` (which is, canonically, a dynamically-sized `bytes` array) and tries to parse it as a conjunction of function signatures.

As is evident in:

code/iexec-solidity/contracts/ERC1538/ERC1538Update.sol:L39

```
if (char == 0x3B) // 0x3B = ';'
```

Inside the function, `;` is being used as a "reserved" character, serving as a delimiter between each function signature.

However, if two semicolons are used in succession, the second one will not be checked and will be made part of the function signature being sent into the `_setFunc()` method.

Example of faulty input

```
someFunc;;someOtherFuncWithSemiColon;
```

Recommendation

Replace the line that increases the `pos` counter at the end of the function:

code/iexec-solidity/contracts/ERC1538/ERC1538Update.sol:L47

```
start = ++pos;
```

WIth this line of code:

```
start = pos + 1;
```

Appendix 1 - Code Quality Recommendations

A.1.1 Use hardcoded hash values instead of constants

Since the Solidity compiler does not yet compute constants which make use of EVM opcodes at compile-time (specifically important for the iExec codebase is the case of the `SHA3` opcode), the audit team recommends that the function signatures and Keccak256 hashes are substituted by hardcoded 4-byte and 32-byte hex values instead. This will result in less deployment and runtime costs overall, with close to no hinderance in auditability.

To create full trust in the hardcoded constants, the dev team may optionally want to verify that the hardcoded constant matches the result of the execution of said opcode by `require()` ing that both the constant and the runtime implementation of the `keccak256()` function with the right parameters match.

Update: iExec team agreed to this suggestion and implemented a fix in PoCo-dev/pull/70/ and d42593966b68524291715662154b1ba436af2be3.

A.1.2 Use of error messages in require()

Given the excessive amount of checks in the codebase (e.g. `matchOrder()`

has 27 explicit require checks), it is suggested to use error messages to simplify debugging and future updates. The full text error messages might result in imploding size of the smart contract, hence it's suggested to add the error message to critical checks and use short error codes instead of (32+ bytes) strings.

Update: iExec team agreed to this suggestion and implemented a partial fix in 3f7f22712821bd5d8cfcf9b279d4af18b0e56bf9. However, error messages increase immensely the deployment size of contracts, effectively rendering them "undeployable". So the fix was only implemented partially.

A.1.3 Variable definitions on top of the contract

In order to have more readable code, it is recommended that all variables are defined on top of the contract code. As an example `Identities` struct is defined in the middle of `IexecPocoDelegate.sol` , and might not be obvious to the reader that there's such definition in that contract.

Update: iExec team agreed to this suggestion and implemented a fix in a number of commits to the repos between April 8, 2020 and April 17, 2020.

A.1.4 Inline documentation increases the code readability

Inline code documentation helps with the code review and most importantly with future code updates. The code base is lacking descriptive comments regarding the decisions of the development team on the implementation. It is suggested to leave the useful code comments when refactoring.

Update: iExec team agreed to this suggestion and implemented a fix in fd91ee07a2bbe3b8eedd65f68ef8271a41960995.

Appendix 2 - Files in Scope

This audit covered the following files in the respective repositories:

iExecBlockchainComputing/poco-dev

File Name	SHA-1 Hash
poco-dev/contracts /IexecInterfaceNative.sol	438599f3acea91f811c7f395 235c1d8a7deda112
poco-dev/contracts /IexecInterfaceNativeABILegacy.sol	28607ea20a6e91fcc5b925b f12f68ff45b96d999
poco-dev/contracts /IexecInterfaceToken.sol	2ea18304e61a6d88a39823a c7136c72e7e0d6256
poco-dev/contracts /IexecInterfaceTokenABILegacy.sol	e0541ee61d54d9034c53d2 9c8c93735a7cc4574f
poco-dev/contracts/Store.sol	b5edb04dabdc5983a117d0 74e7b273e4956fe34f
poco-dev/contracts /libs/IexecLibCore_v5.sol	359c785f15d6ac64197e89a 4f8c358c9eba9ff57
poco-dev/contracts /libs/IexecLibOrders_v5.sol	65d30c4d5069636495034 aa62993516ffcd6b006
poco-dev/contracts/modules /DelegateBase.sol	966321486cf7049912cfaf34 ea8fcfa36a665b09
poco-dev/contracts/modules/delegates /ENSIntegrationDelegate.sol	509ad5bda5fb7896699fe92 fa4f1783f2116453e
poco-dev/contracts/modules/delegates /IexecAccessorsABILegacyDelegate.sol	257f318160dfd6a848c43bfe 2d4db45551398825
poco-dev/contracts/modules/delegates /IexecAccessorsDelegate.sol	8bbc143e3ea0e731c6c5785 689d324c3fc7376a8
poco-dev/contracts/modules/delegates /IexecCategoryManagerDelegate.sol	b42cb5c07838d5eb8da1f8 088e9a1a6e4dac1fb1
poco-dev/contracts/modules/delegates /IexecERC20Common.sol	54ecb31c576017c96fa7e32 2102a039453974f73
poco-dev/contracts/modules/delegates /IexecERC20Delegate.sol	6b6e404844c727e57a13991 c07d90b1b4ed5d05a

File Name	SHA-1 Hash
poco-dev/contracts/modules/delegates/IexecEscrowNativeDelegate.sol	d0f96ed32949a8d072695254eb17acdb8a691337
poco-dev/contracts/modules/delegates/IexecEscrowTokenDelegate.sol	1c0177cff23a426fe40c27d65ab2c854e5cf3cfe
poco-dev/contracts/modules/delegates/IexecMaintenanceDelegate.sol	1c1eef2430cc35ce3366a4ac10fcc9139e845e52
poco-dev/contracts/modules/delegates/IexecMaintenanceExtraDelegate.sol	00b3b7ab05f2f79040200a1528a2f1a5249da606
poco-dev/contracts/modules/delegates/IexecOrderManagementDelegate.sol	aa2f3dccf020d9c21f507701279e92e5c4fc6c79
poco-dev/contracts/modules/delegates/IexecPocoDelegate.sol	a43fa6b7f4c088adfdfe531aceff8e9c73bcc276
poco-dev/contracts/modules/delegates/IexecRelayDelegate.sol	096d24d4b15593ee1cee7f972bd26cb8deab6179
poco-dev/contracts/modules/delegates/SignatureVerifier.sol	83160d2e5924055aa3206f0578c32fc584131ce4
poco-dev/contracts/modules/interfaces/ENSIntegration.sol	f0ad54cfbc0f3f5dda2048af72f81b3b636eaabb
poco-dev/contracts/modules/interfaces/IOwnable.sol	b33a9ad33d580bb88eed1013e13b69835840ef51
poco-dev/contracts/modules/interfaces/IexecAccessors.sol	c2bff677eb8d606af5698adfd8d247cfb7883565
poco-dev/contracts/modules/interfaces/IexecAccessorsABILegacy.sol	91f97256685b91010441f9bf9e51f0e44585a5d5
poco-dev/contracts/modules/interfaces/IexecCategoryManager.sol	2c0bc1c4f9e3261c4e1cee4b78887b14f65b9e1b
poco-dev/contracts/modules/interfaces/IexecERC20.sol	66841034833adca8c16c3011feaac38cd1c768fc

File Name	SHA-1 Hash
poco-dev/contracts/modules/interfaces/IexecEscrowNative.sol	d8847e54490a498845664e05b301ee6a59c2e6dd
poco-dev/contracts/modules/interfaces/IexecEscrowToken.sol	0ff3340f349dd50126d4a7edeebe3417fe7b033e
poco-dev/contracts/modules/interfaces/IexecMaintenance.sol	1822954ab2aa4f315f00547534657fb5e94e5688
poco-dev/contracts/modules/interfaces/IexecMaintenanceExtra.sol	47bdc786183681f4ba0baf29b3d0fcc009eb30bd
poco-dev/contracts/modules/interfaces/IexecOrderManagement.sol	bdc694d099bc20ca89c1577f7b403ce2b0c06b0d
poco-dev/contracts/modules/interfaces/IexecPoco.sol	f82e8e5e5aa70c35345d7a6a318eaa4c0610c246
poco-dev/contracts/modules/interfaces/IexecRelay.sol	be2ab578ba29627be4643efd27598ebd749e7fae
poco-dev/contracts/modules/interfaces/IexecTokenSpender.sol	202b77df4de1fcdacd1a26d0ec72fd0ad96ae720
poco-dev/contracts/registries/IRegistry.sol	ffe3c15f48605d24c5b1497529e01fffc2066b02
poco-dev/contracts/registries/Registry.sol	a3837bdfa95c5024ad1251e60a27c15d76ddefa1
poco-dev/contracts/registries/RegistryEntry.sol	b6864be405a056d6ef172b4a50b30afc35692622
poco-dev/contracts/registries/apps/App.sol	cac8649f11ce8bc2c93b85e003e429b3bce58c0b
poco-dev/contracts/registries/apps/AppRegistry.sol	e1d7c5744cbff24c80dc4b8fd743ed95e1a6e262
poco-dev/contracts/registries/datasets/Dataset.sol	83257f5ac85d8da3460954b2c53fb420b5932390

File Name	SHA-1 Hash
poco-dev/contracts/registries/datasets/DatasetRegistry.sol	bf147967c07446dde52b7b1c275bafaac0644e37
poco-dev/contracts/registries/workerpools/Workerpool.sol	16be9246eb5652d24a46146b541f063ac90be269
poco-dev/contracts/registries/workerpools/WorkerpoolRegistry.sol	cab0ee262cd9d5b42dce9ee6965e540b6b27d1cf
poco-dev/contracts/tools/Migrations.sol	ab396f2c04aed69f6cdef9a954b8f22da7822d21
poco-dev/contracts/tools/testing/TestClient.sol	0bcf03e777105ce8d52d304a3704064ac5a4d944
poco-dev/contracts/tools/testing/TestReceiver.sol	5404782e56839826c5f9649f42f87be409b082c4

iExecBlockchainComputing/iexec-solidity

File Name	SHA-1 Hash
iexec-solidity/contracts/ENStools/ENSReverseRegistration.sol	20ea50fd7ba8fb5398281b34f3ba2172846e1d49
iexec-solidity/contracts/ERC1154/IERC1154.sol	892b56dee343f68a984bdf29d2b25f9f45953630
iexec-solidity/contracts/ERC1271/IERC1271.sol	4944fcc92d2ba5abf07a4aa381f1414859b97fd4
iexec-solidity/contracts/ERC1538/ERC1538.sol	c2ff06da81513e4f0a9143ec4dc03fa0e56d402b
iexec-solidity/contracts/ERC1538/ERC1538Proxy.sol	75e468f9819caace38123ab2934cb936774956f3
iexec-solidity/contracts/ERC1538/ERC1538Query.sol	73f28de88815b08cdeeaea3ad874a8bea677d441
iexec-solidity/contracts/ERC1538/ERC1538Store.sol	6f8bbfd330c5cbb78bc0c74694b3db6b5adce274

File Name	SHA-1 Hash
iexec-solidity/contracts/ERC1538/ERC1538Update.sol	38a9d71ace70289423c577b8ca8931794484a201
iexec-solidity/contracts/ERC1538/IERC1538.sol	2a30f324d44b77a5dda1619c393e1dcc7c45a585
iexec-solidity/contracts/ERC725/IERC725.sol	14e1265d58b916e925300388fff6c4a1b4854c71
iexec-solidity/contracts/ERC734/IERC734.sol	1648464843385275d20db57ba349d78ae95d09af
iexec-solidity/contracts/Factory/CounterfactualFactory.sol	822d7cfba1ca1f2a66304481f59054296e8223f1
iexec-solidity/contracts/Factory/GenericFactory.sol	45888956954bbb2c1a32b60099eeee72a392b135
iexec-solidity/contracts/Libs/SafeMathExtended.sol	988444bcf40be7af53d1485af2f9b8d6d64e27bf
iexec-solidity/contracts/Migrations.sol	d6a9049b9ccf34341831c3d34ea0f8d66dcacea0
iexec-solidity/contracts/TestContract.sol	44e98d4544b0e414281a602975e48f7cc931d85d
iexec-solidity/contracts/Upgradeability/BaseUpgradeabilityProxy.sol	1d7fdce8663c7338ff9ca508be7ef95fcc8a49a1
iexec-solidity/contracts/Upgradeability/InitializableUpgradeabilityProxy.sol	fae44f55f71595c17b7fc6a01da5c7a2e757df3c
iexec-solidity/contracts/Upgradeability/Proxy.sol	a6e3c5967eb838e4a79e763f82d12baaf5db7394

Appendix 3 - Artifacts

This section contains some of the artifacts generated during our review by automated tools, the test suite, etc. If any issues or recommendations were identified by the output presented here, they have been addressed in the

appropriate section above.

A.3.1 MythX

MythX is a security analysis API for Ethereum smart contracts. It performs multiple types of analysis, including fuzzing and symbolic execution, to detect many common vulnerability types. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on MythX can be found at mythx.io.

Below is the miniaturized output of the MythX vulnerability scan per repository. Please note that this does not include multi-contract, multi-transaction issues. Those can only be seen in the tool dashboard but have been analyzed extensively by the audit team.

```
/iexec-solidity/contracts/upgradeability/baseupgradeabilityproxy.sol
  1:0   warning  A floating pragma is set  SWC-103
  8:1   error    integer overflow          SWC-101
  9:41  error    integer overflow          SWC-101

/iexec-solidity/contracts/upgradeability/proxy.sol
   1:0   warning  A floating pragma is set  SWC-103
  47:54  warning  requirement violation     SWC-123
  51:77  warning  requirement violation     SWC-123

/iexec-solidity/contracts/factory/counterfactualfactory.sol
  -1:0   warning  assertion violation
   1:0   warning  A floating pragma is set
  15:11  warning  requirement violation
  28:3   warning  Potentially unbounded data structure passed to builtin

/iexec-solidity/contracts/libs/ecdsa.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-solidity/contracts/libs/ecdsalib.sol
   1:0  warning  A floating pragma is set                      SWC-103
  20:1  warning  The caller can jump to any point in the code  SWC-127

/iexec-solidity/contracts/enstools/ensreverseregistration.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-solidity@ensdomains/ens/contracts/ens.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-solidity/contracts/erc1538/erc1538.sol
  1:0   warning  A floating pragma is set  SWC-103
  6:12  error    integer overflow          SWC-101

/iexec-solidity/contracts/erc1538/erc1538store.sol
   1:0  warning  A floating pragma is set         SWC-103
  10:1  warning  Unused state variable "m_funcs"  SWC-131

/iexec-solidity/contracts/erc1538/ierc1538.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-solidity@openzeppelin/contracts/gsn/context.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-solidity@openzeppelin/contracts/access/ownable.sol
  1:0  warning  A floating pragma is set  SWC-103

/iexec-soliditysolstruct/contracts/libs/libmap2.bytes4.address.bytes.sol
   1:0     warning  A floating pragma is set                     SWC-103
  12:274   warning  Implicit loop over unbounded data structure  SWC-128
```

```
/poco-dev/contracts/registries/iregistry.sol
   3:20  error    persistent state write after call  SWC-107
   3:42  warning  requirement violation              SWC-123
   6:43  error    integer overflow                   SWC-101
  10:20  error    integer overflow                   SWC-101

/poco-dev/contracts/registries/registry.sol
  10:1090  warning  multiple external calls           SWC-113
  10:1090  warning  requirement violation             SWC-123
  10:1159  error    persistent state read after call  SWC-107
  10:1673  warning  requirement violation             SWC-123

/poco-dev/contracts/registries/apps/app.sol
   6:17  error  integer overflow  SWC-101
   6:45  error  integer overflow  SWC-101
   8:22  error  integer overflow  SWC-101
  10:8   error  integer overflow  SWC-101

/poco-dev@iexec/solidity/contracts/enstools/ensreverseregistration.sol
  10:387  warning  Multiple calls are executed in the same transaction  SI
  10:387  warning  requirement violation                                SI
  16:35   warning  requirement violation                                SI

/poco-dev@iexec/solidity/contracts/upgradeability/initializableupgradeabi
  10:719  warning  A reachable exception has been detected  SWC-110
  10:883  warning  requirement violation                    SWC-123

/poco-dev@iexec/solidity/contracts/upgradeability/proxy.sol
  10:1253  warning  requirement violation  SWC-123
  10:1471  warning  requirement violation  SWC-123

/poco-dev@openzeppelin/contracts/token/erc721/erc721.sol
  10:9128   error    persistent state read after call
  10:11878  error    persistent state write after call
  10:11878  error    persistent state read after call
  10:14333  warning  Potentially unbounded data structure passed to built
  10:15790  warning  Unused function parameter "from"
  10:15804  warning  Unused function parameter "to"
  10:15816  warning  Unused function parameter "tokenId"
  72:12641  warning  Unused function parameter "from"
  72:12655  warning  Unused function parameter "to"
  72:12667  warning  Unused function parameter "tokenId"

/poco-dev@openzeppelin/contracts/token/erc721/erc721enumerable.sol
  10:3630  warning  Incorrect function "_tokensOfOwner" state mutability
  10:3721  warning  Implicit loop over unbounded data structure
  10:4132  error    persistent state write after call
  10:4161  error    persistent state read after call
  10:4194  error    persistent state write after call
```

A.3.2 Ethlint

Ethlint is an open source project for linting Solidity code. Only security-related issues were reviewed by the audit team.

Below is the raw output of the Ethlint vulnerability scan per repository.

```
contracts/ENStools/ENSReverseRegistration.sol
  16:1    error    Only use indent of 4 spaces.    indentation
  18:1    error    Only use indent of 4 spaces.    indentation
  22:0    error    Only use indent of 4 spaces.    indentation

contracts/ERC1271/IERC1271.sol
  3:1    error    Syntax error: unexpected token a

contracts/ERC1538/ERC1538.sol
  8:1      error      Only use indent of 4 spaces.
  9:1      error      Only use indent of 4 spaces.
  11:1     error      Only use indent of 4 spaces.
  12:1     error      Only use indent of 4 spaces.
  14:1     error      Only use indent of 4 spaces.
  18:0     error      Only use indent of 4 spaces.
  20:1     error      Only use indent of 4 spaces.
  24:27    warning    There should be no whitespace or comments between tl
  24:56    warning    There should be no whitespace or comments between tl
  25:27    warning    There should be no whitespace or comments between tl
  25:56    warning    There should be no whitespace or comments between tl
  43:0     error      Only use indent of 4 spaces.

contracts/ERC1538/ERC1538Proxy.sol
  9:1     error    Only use indent of 4 spaces.    indentation
  10:1    error    Only use indent of 4 spaces.    indentation
  12:1    error    Only use indent of 4 spaces.    indentation
  18:0    error    Only use indent of 4 spaces.    indentation
  20:1    error    Only use indent of 4 spaces.    indentation
  25:0    error    Only use indent of 4 spaces.    indentation

contracts/ERC1538/ERC1538ProxyV2.sol
  9:1     error    Only use indent of 4 spaces.    indentation
  10:1    error    Only use indent of 4 spaces.    indentation
  12:1    error    Only use indent of 4 spaces.    indentation
  18:0    error    Only use indent of 4 spaces.    indentation
  20:1    error    Only use indent of 4 spaces.    indentation
  25:0    error    Only use indent of 4 spaces.    indentation

contracts/ERC1538/ERC1538Query.sol
  20:1     error    Only use indent of 4 spaces.                    inde
  24:0     error    Only use indent of 4 spaces.                    inde
  26:1     error    Only use indent of 4 spaces.                    inde
  31:0     error    Only use indent of 4 spaces.                    inde
  33:1     error    Only use indent of 4 spaces.                    inde
  37:0     error    Only use indent of 4 spaces.                    inde
  39:1     error    Only use indent of 4 spaces.                    inde
  43:0     error    Only use indent of 4 spaces.                    inde
  45:1     error    Only use indent of 4 spaces.                    inde
  49:0     error    Only use indent of 4 spaces.                    inde
```

```
contracts/IexecInterfaceNative.sol
  17:32    error    Syntax error: unexpected token i

contracts/IexecInterfaceNativeABILegacy.sol
  18:41    error    Syntax error: unexpected token i

contracts/IexecInterfaceToken.sol
  17:31    error    Syntax error: unexpected token i

contracts/IexecInterfaceTokenABILegacy.sol
  18:40    error    Syntax error: unexpected token i

contracts/Store.sol
  24:1    error    Syntax error: unexpected token a

contracts/libs/IexecLibCore_v5.sol
  9:1      error    Only use indent of 4 spaces.    indentation
  13:0     error    Only use indent of 4 spaces.    indentation
  14:1     error    Only use indent of 4 spaces.    indentation
  19:0     error    Only use indent of 4 spaces.    indentation
  24:1     error    Only use indent of 4 spaces.    indentation
  29:0     error    Only use indent of 4 spaces.    indentation
  30:1     error    Only use indent of 4 spaces.    indentation
  51:0     error    Only use indent of 4 spaces.    indentation
  56:1     error    Only use indent of 4 spaces.    indentation
  63:0     error    Only use indent of 4 spaces.    indentation
  64:1     error    Only use indent of 4 spaces.    indentation
  80:0     error    Only use indent of 4 spaces.    indentation
  85:1     error    Only use indent of 4 spaces.    indentation
  89:0     error    Only use indent of 4 spaces.    indentation
  94:1     error    Only use indent of 4 spaces.    indentation
  100:0    error    Only use indent of 4 spaces.    indentation
  101:1    error    Only use indent of 4 spaces.    indentation
  108:0    error    Only use indent of 4 spaces.    indentation

contracts/libs/IexecLibOrders_v5.sol
  7:1      warning    Line exceeds the limit of 145 characters
  7:1      error      Only use indent of 4 spaces.
  8:1      error      Only use indent of 4 spaces.
  8:1      warning    Line exceeds the limit of 145 characters
  9:1      warning    Line exceeds the limit of 145 characters
  9:1      error      Only use indent of 4 spaces.
  10:1     warning    Line exceeds the limit of 145 characters
  10:1     error      Only use indent of 4 spaces.
  11:1     error      Only use indent of 4 spaces.
  11:1     warning    Line exceeds the limit of 145 characters
  12:1     warning    Line exceeds the limit of 145 characters
  12:1     error      Only use indent of 4 spaces.
  13:1     error      Only use indent of 4 spaces.
```

A.3.3 Surya

Surya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Below is the tool output per repository.

Sūrya's Description Report For The `iexec-solidity` Repository

File Name	SHA-1 Hash
iexec-solidity/contracts/ENStools/ENSReverseRegistration.sol	20ea50fd7ba8fb5398281b34f3ba2172846e1d49
iexec-solidity/contracts/ERC1154/IERC1154.sol	892b56dee343f68a984bdf29d2b25f9f45953630
iexec-solidity/contracts/ERC1271/IERC1271.sol	4944fcc92d2ba5abf07a4aa381f1414859b97fd4
iexec-solidity/contracts/ERC1538/ERC1538.sol	c2ff06da81513e4f0a9143ec4dc03fa0e56d402b
iexec-solidity/contracts/ERC1538/ERC1538Proxy.sol	75e468f9819caace38123ab2934cb936774956f3
iexec-solidity/contracts/ERC1538/ERC1538ProxyV2.sol	05a295a9c62eda7d6c106174a36cfc87dc446107
iexec-solidity/contracts/ERC1538/ERC1538Query.sol	73f28de88815b08cdeeaea3ad874a8bea677d441
iexec-solidity/contracts/ERC1538/ERC1538Store.sol	6f8bbfd330c5cbb78bc0c74694b3db6b5adce274
iexec-solidity/contracts/ERC1538/ERC1538Update.sol	38a9d71ace70289423c577b8ca8931794484a201
iexec-solidity/contracts/ERC1538/ERC1538UpdateV2.sol	6830163504f53c40271a7a51e515421d62d2137b

File Name	SHA-1 Hash
iexec-solidity/contracts/ERC1538/IERC1538.sol	2a30f324d44b77a5dda1619c393e1dcc7c45a585
iexec-solidity/contracts/ERC725/IERC725.sol	14e1265d58b916e925300388fff6c4a1b4854c71
iexec-solidity/contracts/ERC734/IERC734.sol	1648464843385275d20db57ba349d78ae95d09af
iexec-solidity/contracts/Factory/CounterfactualFactory.sol	822d7cfba1ca1f2a66304481f59054296e8223f1
iexec-solidity/contracts/Factory/GenericFactory.sol	45888956954bbb2c1a32b60099eeee72a392b135
iexec-solidity/contracts/Libs/ECDSA.sol	3fa8517670e83c2219c5c0ead6416233e6c03c20
iexec-solidity/contracts/Libs/ECDSALib.sol	ea8e62fa6f1f489ecc26f156f603dfb1ffe6ba9d
iexec-solidity/contracts/Libs/SafeMathExtended.sol	988444bcf40be7af53d1485af2f9b8d6d64e27bf
iexec-solidity/contracts/Libs/SignatureVerifier.sol	54f3d4406e3998d6effe31b9366d71460229cdda
iexec-solidity/contracts/Migrations.sol	d6a9049b9ccf34341831c3d34ea0f8d66dcacea0
iexec-solidity/contracts/TestContract.sol	44e98d4544b0e414281a602975e48f7cc931d85d
iexec-solidity/contracts/Upgradeability/BaseUpgradeabilityProxy.sol	1d7fdce8663c7338ff9ca508be7ef95fcc8a49a1
iexec-solidity/contracts/Upgradeability/InitializableUpgradeabilityProxy.sol	fae44f55f71595c17b7fc6a01da5c7a2e757df3c
iexec-solidity/contracts/Upgradeability/Proxy.sol	a6e3c5967eb838e4a79e763f82d12baaf5db7394

Contract	Type	Bases		

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IReverseRegistrar	Interface			
L	claim	External ❗	🛑	NO ❗
L	claimWithResolver	External ❗	🛑	NO ❗
L	setName	External ❗	🛑	NO ❗
L	node	External ❗		NO ❗
ENSReverseRegistration	Implementation			
L	_setName	Internal 🔒	🛑	
IOracleConsumer	Interface			
L	receiveResult	External ❗	🛑	NO ❗
IOracle	Interface			
L	resultFor	External ❗		NO ❗
IERC1271	Implementation			
L	isValidSignature	Public ❗		NO ❗
ERC1538	Implementation	IERC1538, ERC1538Store		
L		Public ❗	🛑	NO ❗

Contract	Type	Bases		
L	_setFunc	Internal 🔒	🛑	
ERC1538Proxy	Implementation	ERC1538, Proxy		
L		Public ❗	🛑	NO ❗
L	_implementation	Internal 🔒		
ERC1538ProxyV2	Implementation	ERC1538, Proxy		
L		Public ❗	🛑	NO ❗
L	_implementation	Internal 🔒		
ERC1538Query	Interface			
L	totalFunctions	External ❗		NO ❗
L	functionByIndex	External ❗		NO ❗
L	functionById	External ❗		NO ❗
L	functionExists	External ❗		NO ❗
L	functionSignatures	External ❗		NO ❗
L	delegateFunctionSignatures	External ❗		NO ❗
L	delegateAddress	External ❗		NO ❗

Contract	Type	Bases		
L	delegateAddresses	External ❗		NO ❗
ERC1538QueryDelegate	Implementation	ERC1538Query, ERC1538		
L	totalFunctions	External ❗		NO ❗
L	functionByIndex	External ❗		NO ❗
L	functionById	External ❗		NO ❗
L	functionExists	External ❗		NO ❗
L	delegateAddress	External ❗		NO ❗
L	functionSignatures	External ❗		NO ❗
L	delegateFunctionSignatures	External ❗		NO ❗
L	delegateAddresses	External ❗		NO ❗
ERC1538Store	Implementation	Ownable		
ERC1538Update	Interface			
L	updateContract	External ❗	🛑	NO ❗

Contract	Type	Bases		
ERC1538UpdateDelegate	Implementation	ERC1538Update, ERC1538		
L	updateContract	External ❗	🛑	onlyOwner
ERC1538UpdateV2	Interface			
L	updateContract	External ❗	🛑	NO ❗
ERC1538UpdateV2Delegate	Implementation	ERC1538UpdateV2, ERC1538		
L	updateContract	External ❗	🛑	onlyOwner
IERC1538	Interface			
IERC725	Interface			
L	getData	External ❗		NO ❗
L	setData	External ❗	🛑	NO ❗
L	execute	External ❗	🛑	NO ❗
IERC734	Implementation			
L	getKey	External ❗		NO ❗
L	keyHasPurpose	External ❗		NO ❗
L	getKeysByPurpose	External ❗		NO ❗

Contract	Type	Bases		
L	addKey	External ❗	🛑	NO ❗
L	removeKey	External ❗	🛑	NO ❗
L	execute	External ❗	🛑	NO ❗
L	approve	External ❗	🛑	NO ❗
CounterfactualFactory	Implementation			
L	_create2	Internal 🔒	🛑	
L	_predictAddress	Internal 🔒		
GenericFactory	Implementation	CounterfactualFactory		
L	predictAddress	Public ❗		NO ❗
L	createContract	Public ❗	🛑	NO ❗
L	predictAddressWithCall	Public ❗		NO ❗
L	createContractAndCall	Public ❗	🛑	NO ❗
ECDSA	Implementation			
L	recover	Internal 🔒		
L	recover	Internal 🔒		
L	toEthSignedMessageHash	Internal 🔒		

Contract	Type	Bases		
L	toEthTyped StructHash	Internal 🔒		
ECDSALib	Library			
L	recover	Public ❗		NO ❗
L	recover	Public ❗		NO ❗
L	toEthSigned MessageHash	Public ❗		NO ❗
L	toEthTyped StructHash	Public ❗		NO ❗
SafeMathExtended	Library			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
L	max	Internal 🔒		
L	min	Internal 🔒		
L	mulByFraction	Internal 🔒		
L	percentage	Internal 🔒		
L	log	Internal 🔒		
SignatureVerifier	Implementation	ECDSA		
L	_isContract	Internal 🔒		

Contract	Type	Bases		
L	_addrToKey	Internal 🔒		
L	_checkIdentity	Internal 🔒		
L	_checkSignature	Internal 🔒		
Migrations	Implementation			
L		Public ❗	🛑	NO ❗
L	setCompleted	Public ❗	🛑	restricted
L	upgrade	Public ❗	🛑	restricted
TestContract	Implementation			
L		External ❗	💵	NO ❗
L		External ❗	💵	NO ❗
L	set	External ❗	🛑	NO ❗
BaseUpgradeabilityProxy	Implementation	Proxy		
L	_implementation	Internal 🔒		
L	_upgradeTo	Internal 🔒	🛑	
L	_setImplementation	Internal 🔒	🛑	
InitializableUpgradeabilityProxy	Implementation	BaseUpgradeabilityProxy		

Contract	Type	Bases		
L	initialize	Public ❗	💵	NO ❗
Proxy	Implementation			
L		External ❗	💵	NO ❗
L		External ❗	💵	NO ❗
L	_implementation	Internal 🔒		
L	_delegate	Internal 🔒	🛑	
L	_willFallback	Internal 🔒	🛑	
L	_fallback	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Sūrya's Description Report For The `poco-dev` Repository

File Name	SHA-1 Hash
poco-dev/contracts/IexecInterfaceNative.sol	438599f3acea91f811c7f395235c1d8a7deda112
poco-dev/contracts/IexecInterfaceNativeABILegacy.sol	28607ea20a6e91fcc5b925bf12f68ff45b96d999
poco-dev/contracts/IexecInterfaceToken.sol	2ea18304e61a6d88a39823ac7136c72e7e0d6256
poco-dev/contracts/IexecInterfaceTokenABILegacy.sol	e0541ee61d54d9034c53d29c8c93735a7cc4574f

File Name	SHA-1 Hash
poco-dev/contracts/Store.sol	b5edb04dabdc5983a117d074e7b273e4956fe34f
poco-dev/contracts/libs/IexecLibCore_v5.sol	359c785f15d6ac64197e89a4f8c358c9eba9ff57
poco-dev/contracts/libs/IexecLibOrders_v5.sol	65d30c4d5069636495034aa62993516ffcd6b006
poco-dev/contracts/modules/DelegateBase.sol	966321486cf7049912cfaf34ea8fcfa36a665b09
poco-dev/contracts/modules/delegates/ENSIntegrationDelegate.sol	509ad5bda5fb7896699fe92fa4f1783f2116453e
poco-dev/contracts/modules/delegates/IexecAccessorsABILegacyDelegate.sol	257f318160dfd6a848c43bfe2d4db45551398825
poco-dev/contracts/modules/delegates/IexecAccessorsDelegate.sol	8bbc143e3ea0e731c6c5785689d324c3fc7376a8
poco-dev/contracts/modules/delegates/IexecCategoryManagerDelegate.sol	b42cb5c07838d5eb8da1f8088e9a1a6e4dac1fb1
poco-dev/contracts/modules/delegates/IexecERC20Common.sol	54ecb31c576017c96fa7e322102a039453974f73
poco-dev/contracts/modules/delegates/IexecERC20Delegate.sol	6b6e404844c727e57a13991c07d90b1b4ed5d05a
poco-dev/contracts/modules/delegates/IexecEscrowNativeDelegate.sol	d0f96ed32949a8d072695254eb17acdb8a691337
poco-dev/contracts/modules/delegates/IexecEscrowTokenDelegate.sol	1c0177cff23a426fe40c27d65ab2c854e5cf3cfe
poco-dev/contracts/modules/delegates/IexecMaintenanceDelegate.sol	1c1eef2430cc35ce3366a4ac10fcc9139e845e52
poco-dev/contracts/modules/delegates/IexecMaintenanceExtraDelegate.sol	00b3b7ab05f2f79040200a1528a2f1a5249da606

File Name	SHA-1 Hash
poco-dev/contracts/modules/delegates/IexecOrderManagementDelegate.sol	aa2f3dccf020d9c21f507701279e92e5c4fc6c79
poco-dev/contracts/modules/delegates/IexecPocoDelegate.sol	a43fa6b7f4c088adfdfe531aceff8e9c73bcc276
poco-dev/contracts/modules/delegates/IexecRelayDelegate.sol	096d24d4b15593ee1cee7f972bd26cb8deab6179
poco-dev/contracts/modules/delegates/SignatureVerifier.sol	83160d2e5924055aa3206f0578c32fc584131ce4
poco-dev/contracts/modules/interfaces/ENSIntegration.sol	f0ad54cfbc0f3f5dda2048af72f81b3b636eaabb
poco-dev/contracts/modules/interfaces/IOwnable.sol	b33a9ad33d580bb88eed1013e13b69835840ef51
poco-dev/contracts/modules/interfaces/IexecAccessors.sol	c2bff677eb8d606af5698adfd8d247cfb7883565
poco-dev/contracts/modules/interfaces/IexecAccessorsABILegacy.sol	91f97256685b91010441f9bf9e51f0e44585a5d5
poco-dev/contracts/modules/interfaces/IexecCategoryManager.sol	2c0bc1c4f9e3261c4e1cee4b78887b14f65b9e1b
poco-dev/contracts/modules/interfaces/IexecERC20.sol	66841034833adca8c16c3011feaac38cd1c768fc
poco-dev/contracts/modules/interfaces/IexecEscrowNative.sol	d8847e54490a498845664e05b301ee6a59c2e6dd
poco-dev/contracts/modules/interfaces/IexecEscrowToken.sol	0ff3340f349dd50126d4a7edeebe3417fe7b033e
poco-dev/contracts/modules/interfaces/IexecMaintenance.sol	1822954ab2aa4f315f00547534657fb5e94e5688
poco-dev/contracts/modules/interfaces/IexecMaintenanceExtra.sol	47bdc786183681f4ba0baf29b3d0fcc009eb30bd

File Name	SHA-1 Hash
poco-dev/contracts/modules/interfaces/IexecOrderManagement.sol	bdc694d099bc20ca89c1577f7b403ce2b0c06b0d
poco-dev/contracts/modules/interfaces/IexecPoco.sol	f82e8e5e5aa70c35345d7a6a318eaa4c0610c246
poco-dev/contracts/modules/interfaces/IexecRelay.sol	be2ab578ba29627be4643efd27598ebd749e7fae
poco-dev/contracts/modules/interfaces/IexecTokenSpender.sol	202b77df4de1fcdacd1a26d0ec72fd0ad96ae720
poco-dev/contracts/registries/IRegistry.sol	ffe3c15f48605d24c5b1497529e01fffc2066b02
poco-dev/contracts/registries/Registry.sol	a3837bdfa95c5024ad1251e60a27c15d76ddefa1
poco-dev/contracts/registries/RegistryEntry.sol	b6864be405a056d6ef172b4a50b30afc35692622
poco-dev/contracts/registries/apps/App.sol	cac8649f11ce8bc2c93b85e003e429b3bce58c0b
poco-dev/contracts/registries/apps/AppRegistry.sol	e1d7c5744cbff24c80dc4b8fd743ed95e1a6e262
poco-dev/contracts/registries/datasets/Dataset.sol	83257f5ac85d8da3460954b2c53fb420b5932390
poco-dev/contracts/registries/datasets/DatasetRegistry.sol	bf147967c07446dde52b7b1c275bafaac0644e37
poco-dev/contracts/registries/workerpools/Workerpool.sol	16be9246eb5652d24a46146b541f063ac90be269
poco-dev/contracts/registries/workerpools/WorkerpoolRegistry.sol	cab0ee262cd9d5b42dce9ee6965e540b6b27d1cf
poco-dev/contracts/tools/Migrations.sol	ab396f2c04aed69f6cdef9a954b8f22da7822d21

File Name	SHA-1 Hash
poco-dev/contracts/tools/testing/TestClient.sol	0bcf03e777105ce8d52d304a3704064ac5a4d944
poco-dev/contracts/tools/testing/TestReceiver.sol	5404782e56839826c5f9649f42f87be409b082c4

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IexecInterfaceNative	Interface	IOwnable, IexecAccessors, IexecCategoryManager, IexecERC20, IexecEscrowNative, IexecMaintenance, IexecOrderManagement, IexecPoco, IexecRelay, IexecTokenSpender, ENSIntegration		
IexecInterfaceNativeABILegacy	Interface	IOwnable, IexecAccessors, IexecAccessorsABILegacy, IexecCategoryManager, IexecERC20,		

Contract	Type	Bases		
		IexecEscrowNative, IexecMaintenance, IexecOrderManagement, IexecPoco, IexecRelay, IexecTokenSpender, ENSIntegration		
IexecInterfaceToken	Interface	IOwnable, IexecAccessors, IexecCategoryManager, IexecERC20, IexecEscrowToken, IexecMaintenance, IexecOrderManagement, IexecPoco, IexecRelay, IexecTokenSpender, ENSIntegration		
IexecInterfaceTokenABILegacy	Interface	IOwnable, IexecAccessors, IexecAccesso		

Contract	Type	Bases		
		rsABILegacy, IexecCategoryManager, IexecERC20, IexecEscrowToken, IexecMaintenance, IexecOrderManagement, IexecPoco, IexecRelay, IexecTokenSpender, ENSIntegration		
Store	Implementation	ERC1538Store		
IexecLibCore_v5	Library			
IexecLibOrders_v5	Library			
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗

Contract	Type	Bases		
L	hash	Public ❗		NO ❗
L	hash	Public ❗		NO ❗
L	toEthSignedMessageHash	Public ❗		NO ❗
L	toEthTypedStructHash	Public ❗		NO ❗
L	recover	Public ❗		NO ❗
DelegateBase	Implementation	Store		
L		Internal 🔒	🛑	
ENSIntegrationDelegate	Implementation	ENSIntegration, ENSReverseRegistration, DelegateBase		
L	setName	External ❗	🛑	onlyOwner
IexecAccessorsABILegacyDelegate	Implementation	IexecAccessorsABILegacy, DelegateBase		
L	viewDealABILegacy_pt1	External ❗		NO ❗
L	viewDealABILegacy_pt2	External ❗		NO ❗
L	viewConfigABILegacy	External ❗		NO ❗

Contract	Type	Bases		
L	viewAccountABILegacy	External ❗		NO ❗
L	viewTaskABILegacy	External ❗		NO ❗
L	viewContributionABILegacy	External ❗		NO ❗
L	viewCategoryABILegacy	External ❗		NO ❗
IexecAccessorsDelegate	Implementation	IexecAccessors, DelegateBase		
L	name	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	frozenOf	External ❗		NO ❗
L	allowance	External ❗		NO ❗
L	viewAccount	External ❗		NO ❗
L	token	External ❗		NO ❗
L	viewDeal	External ❗		NO ❗
L	viewConsumed	External ❗		NO ❗

Contract	Type	Bases		
L	viewPresigned	External ❗		NO ❗
L	viewTask	External ❗		NO ❗
L	viewContribution	External ❗		NO ❗
L	viewScore	External ❗		NO ❗
L	resultFor	External ❗		NO ❗
L	viewCategory	External ❗		NO ❗
L	countCategory	External ❗		NO ❗
L	appregistry	External ❗		NO ❗
L	datasetregistry	External ❗		NO ❗
L	workerpool registry	External ❗		NO ❗
L	teebroker	External ❗		NO ❗
L	callbackgas	External ❗		NO ❗
L	contribution_deadline_ratio	External ❗		NO ❗
L	reveal_deadline_ratio	External ❗		NO ❗
L	final_deadline_ratio	External ❗		NO ❗
L	workerpool_stake_rati	External ❗		NO ❗

Contract	Type	Bases		
	o			
L	kitty_ratio	External ❗		NO ❗
L	kitty_min	External ❗		NO ❗
L	kitty_address	External ❗		NO ❗
L	groupmember_purpose	External ❗		NO ❗
L	eip712domain_separator	External ❗		NO ❗
IexecCategoryManagerDelegate	Implementation	IexecCategoryManager, DelegateBase		
L	createCategory	External ❗	🛑	onlyOwner
IexecERC20Common	Implementation	DelegateBase		
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
IexecERC20Delegate	Implementation	IexecERC20, DelegateBase, IexecERC20Common		

Contract	Type	Bases		
L	transfer	External ❗	🛑	NO ❗
L	approve	External ❗	🛑	NO ❗
L	approveAndCall	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
L	increaseAllowance	External ❗	🛑	NO ❗
L	decreaseAllowance	External ❗	🛑	NO ❗
IexecEscrowNativeDelegate	Implementation	IexecEscrowNative, DelegateBase, IexecERC20Common		
L		External ❗	💵	NO ❗
L	deposit	External ❗	💵	NO ❗
L	depositFor	External ❗	💵	NO ❗
L	depositForArray	External ❗	💵	NO ❗
L	withdraw	External ❗	🛑	NO ❗
L	recover	External ❗	🛑	onlyOwner
L	_deposit	Internal 🔒	🛑	
L	_withdraw	Internal 🔒	🛑	
IexecEscrowTokenDelegate	Implementation	IexecEscrowToken, IexecTokenSp		

Contract	Type	Bases		
		ender, DelegateBase, IexecERC20Common		
L		External ❗	💵	NO ❗
L	deposit	External ❗	🛑	NO ❗
L	depositFor	External ❗	🛑	NO ❗
L	depositForArray	External ❗	🛑	NO ❗
L	withdraw	External ❗	🛑	NO ❗
L	recover	External ❗	🛑	onlyOwner
L	receiveApproval	External ❗	🛑	NO ❗
L	_deposit	Internal 🔒	🛑	
L	_withdraw	Internal 🔒	🛑	
IexecMaintenanceDelegate	Implementation	IexecMaintenance, DelegateBase		
L	configure	External ❗	🛑	onlyOwner
L	domain	External ❗		NO ❗
L	updateDomainSeparator	External ❗	🛑	NO ❗
L	importScore	External ❗	🛑	NO ❗
L	setTeeBroker	External ❗	🛑	onlyOwner

Contract	Type	Bases		
L	setCallbackGas	External ❗	🛑	onlyOwner
L	_chainId	Internal 🔒		
L	_domain	Internal 🔒		
IexecMaintenanceExtraDelegate	Implementation	IexecMaintenanceExtra, DelegateBase		
L	changeRegistries	External ❗	🛑	onlyOwner
IexecOrderManagementDelegate	Implementation	IexecOrderManagement, DelegateBase		
L	manageAppOrder	Public ❗	🛑	NO ❗
L	manageDatasetOrder	Public ❗	🛑	NO ❗
L	manageWorkerpoolOrder	Public ❗	🛑	NO ❗
L	manageRequestOrder	Public ❗	🛑	NO ❗

Contract	Type	Bases		
IexecPoco Delegate	Implementation	IexecPoco, DelegateBase, IexecERC20Common, SignatureVerifier		
L	reward	Internal 🔒	🛑	
L	seize	Internal 🔒	🛑	
L	lock	Internal 🔒	🛑	
L	unlock	Internal 🔒	🛑	
L	lockContribution	Internal 🔒	🛑	
L	unlockContribution	Internal 🔒	🛑	
L	rewardForContribution	Internal 🔒	🛑	
L	seizeContribution	Internal 🔒	🛑	
L	rewardForScheduling	Internal 🔒	🛑	
L	successWork	Internal 🔒	🛑	
L	failedWork	Internal 🔒	🛑	
L	verifySignature	External ❗		NO ❗
L	verifyPresignature	External ❗		NO ❗

Contract	Type	Bases		
L	verifyPresignatureOrSignature	External ❗		NO ❗
L	matchOrders	Public ❗	🛑	NO ❗
L	initialize	Public ❗	🛑	NO ❗
L	contribute	Public ❗	🛑	NO ❗
L	reveal	External ❗	🛑	NO ❗
L	reopen	External ❗	🛑	onlyScheduler
L	finalize	External ❗	🛑	onlyScheduler
L	claim	Public ❗	🛑	NO ❗
L	contributeAndFinalize	Public ❗	🛑	NO ❗
L	checkConsensus	Internal 🔒	🛑	
L	distributeRewards	Internal 🔒	🛑	
L	executeCallback	Internal 🔒	🛑	
L	initializeArray	External ❗	🛑	NO ❗
L	claimArray	External ❗	🛑	NO ❗
L	initializeAndClaimArray	External ❗	🛑	NO ❗

Contract	Type	Bases		
IexecRelayDelegate	Implementation	IexecRelay, DelegateBase		
L	broadcastAppOrder	External ❗	🛑	NO ❗
L	broadcastDatasetOrder	External ❗	🛑	NO ❗
L	broadcastWorkerpoolOrder	External ❗	🛑	NO ❗
L	broadcastRequestOrder	External ❗	🛑	NO ❗
SignatureVerifier	Implementation	DelegateBase		
L	_isContract	Internal 🔒		
L	_addrToKey	Internal 🔒		
L	_checkIdentity	Internal 🔒		
L	_checkSignature	Internal 🔒		
L	_checkPresignature	Internal 🔒		
L	_checkPresignatureOrSignature	Internal 🔒		
ENSIntegration	Interface			
L	setName	External ❗	🛑	NO ❗

Contract	Type	Bases		
IOwnable	Interface			
L	owner	External ❗		NO ❗
L	renounceOwnership	External ❗	🛑	NO ❗
L	transferOwnership	External ❗	🛑	NO ❗
IexecAccessors	Interface	IOracle		
L	name	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	frozenOf	External ❗		NO ❗
L	allowance	External ❗		NO ❗
L	viewAccount	External ❗		NO ❗
L	token	External ❗		NO ❗
L	viewDeal	External ❗		NO ❗
L	viewConsumed	External ❗		NO ❗
L	viewPresigned	External ❗		NO ❗
L	viewTask	External ❗		NO ❗
L	viewContri	External ❗		NO ❗

Contract	Type	Bases		
	bution			
L	viewScore	External ❗		NO ❗
L	viewCategory	External ❗		NO ❗
L	countCategory	External ❗		NO ❗
L	appregistry	External ❗		NO ❗
L	datasetregistry	External ❗		NO ❗
L	workerpool registry	External ❗		NO ❗
L	teebroker	External ❗		NO ❗
L	callbackgas	External ❗		NO ❗
L	contribution_deadline_ratio	External ❗		NO ❗
L	reveal_deadline_ratio	External ❗		NO ❗
L	final_deadline_ratio	External ❗		NO ❗
L	workerpool_stake_ratio	External ❗		NO ❗
L	kitty_ratio	External ❗		NO ❗
L	kitty_min	External ❗		NO ❗
L	kitty_address	External ❗		NO ❗

Contract	Type	Bases		
L	groupmember_purpose	External ❗		NO ❗
L	eip712domain_separator	External ❗		NO ❗
IexecAccessorsABILegacy	Interface			
L	viewAccountABILegacy	External ❗		NO ❗
L	viewDealABILegacy_pt1	External ❗		NO ❗
L	viewDealABILegacy_pt2	External ❗		NO ❗
L	viewTaskABILegacy	External ❗		NO ❗
L	viewContributionABILegacy	External ❗		NO ❗
L	viewCategoryABILegacy	External ❗		NO ❗
L	viewConfigABILegacy	External ❗		NO ❗
IexecCategoryMana	Interface			

Contract	Type	Bases		
ger				
L	createCategory	External ❗	🛑	NO ❗
IexecERC20	Interface			
L	transfer	External ❗	🛑	NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
L	increaseAllowance	External ❗	🛑	NO ❗
L	decreaseAllowance	External ❗	🛑	NO ❗
L	approveAndCall	External ❗	🛑	NO ❗
IexecEscrowNative	Interface			
L		External ❗	💵	NO ❗
L	deposit	External ❗	💵	NO ❗
L	depositFor	External ❗	💵	NO ❗
L	depositForArray	External ❗	💵	NO ❗
L	withdraw	External ❗	🛑	NO ❗
L	recover	External ❗	🛑	NO ❗
IexecEscrowToken	Interface			

Contract	Type	Bases		
L		External ❗	💵	NO ❗
L	deposit	External ❗	🛑	NO ❗
L	depositFor	External ❗	🛑	NO ❗
L	depositForArray	External ❗	🛑	NO ❗
L	withdraw	External ❗	🛑	NO ❗
L	recover	External ❗	🛑	NO ❗
IexecMaintenance	Interface			
L	configure	External ❗	🛑	NO ❗
L	domain	External ❗		NO ❗
L	updateDomainSeparator	External ❗	🛑	NO ❗
L	importScore	External ❗	🛑	NO ❗
L	setTeeBroker	External ❗	🛑	NO ❗
L	setCallbackGas	External ❗	🛑	NO ❗
IexecMaintenanceExtra	Interface			
L	changeRegistries	External ❗	🛑	NO ❗
IexecOrderManagem	Interface			

Contract	Type	Bases		
ent				
L	manageAppOrder	External ❗	🛑	NO ❗
L	manageDatasetOrder	External ❗	🛑	NO ❗
L	manageWorkerpoolOrder	External ❗	🛑	NO ❗
L	manageRequestOrder	External ❗	🛑	NO ❗
IexecPoco	Interface			
L	verifySignature	External ❗		NO ❗
L	verifyPresignature	External ❗		NO ❗
L	verifyPresignatureOrSignature	External ❗		NO ❗
L	matchOrders	External ❗	🛑	NO ❗
L	initialize	External ❗	🛑	NO ❗
L	contribute	External ❗	🛑	NO ❗
L	reveal	External ❗	🛑	NO ❗
L	reopen	External ❗	🛑	NO ❗
L	finalize	External ❗	🛑	NO ❗
L	claim	External ❗	🛑	NO ❗

Contract	Type	Bases		
L	contributeAndFinalize	External ❗	🛑	NO ❗
L	initializeArray	External ❗	🛑	NO ❗
L	claimArray	External ❗	🛑	NO ❗
L	initializeAndClaimArray	External ❗	🛑	NO ❗
IexecRelay	Interface			
L	broadcastAppOrder	External ❗	🛑	NO ❗
L	broadcastDatasetOrder	External ❗	🛑	NO ❗
L	broadcastWorkerpoolOrder	External ❗	🛑	NO ❗
L	broadcastRequestOrder	External ❗	🛑	NO ❗
IexecTokenSpender	Interface			
L	receiveApproval	External ❗	🛑	NO ❗
IRegistry	Implementation	IERC721Enumerable		
L	isRegistered	External ❗		NO ❗

Contract	Type	Bases		
Registry	Implementation	IRegistry, ERC721Full, ENSReverseRegistration, Ownable		
L		Public ❗	🛑	ERC721Full
L	initialize	External ❗	🛑	onlyOwner
L	_mintCreate	Internal 🔒	🛑	
L	_mintPredict	Internal 🔒		
L	isRegistered	External ❗		NO ❗
L	setName	External ❗	🛑	onlyOwner
L	setTokenURI	External ❗	🛑	NO ❗
RegistryEntry	Implementation	ENSReverseRegistration		
L	_initialize	Internal 🔒	🛑	
L	owner	Public ❗		NO ❗
L	setName	External ❗	🛑	onlyOwner
App	Implementation	RegistryEntry		
L	initialize	Public ❗	🛑	NO ❗
AppRegistry	Implementation	Registry		
L		Public ❗	🛑	Registry

Contract	Type	Bases		
L	encodeInitializer	Internal 🔒		
L	createApp	External ❗	🛑	NO ❗
L	predictApp	External ❗		NO ❗
Dataset	Implementation	RegistryEntry		
L	initialize	Public ❗	🛑	NO ❗
DatasetRegistry	Implementation	Registry		
L		Public ❗	🛑	Registry
L	encodeInitializer	Internal 🔒		
L	createDataset	External ❗	🛑	NO ❗
L	predictDataset	External ❗		NO ❗
Workerpool	Implementation	RegistryEntry		
L	initialize	Public ❗	🛑	NO ❗
L	changePolicy	External ❗	🛑	onlyOwner
WorkerpoolRegistry	Implementation	Registry		
L		Public ❗	🛑	Registry
L	encodeInitializer	Internal 🔒		

Contract	Type	Bases		
L	createWorkerpool	External ❗	🛑	NO ❗
L	predictWorkerpool	External ❗		NO ❗
Migrations	Implementation	Ownable		
L		Public ❗	🛑	NO ❗
L	setCompleted	Public ❗	🛑	onlyOwner
L	upgrade	Public ❗	🛑	onlyOwner
TestClient	Implementation	IOracleConsumer		
L		Public ❗	🛑	NO ❗
L	receiveResult	External ❗	🛑	NO ❗
TestReceiver	Implementation	IexecTokenSpender		
L		Public ❗	🛑	NO ❗
L	receiveApproval	External ❗	🛑	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

A.3.4 Tests Suite

Below is the output generated by running the test suite per repository.

```
Started ganache daemon (pid=33197)
Compiling ... success
Migrating ... success
Running tests ... Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

# web3 version: 1.2.1
Chainid is: 1589476946750
Chaintype is: private


  Contract: ERC1538
    ✓ Ownership (100ms)
    ✓ ERC1538Query - totalFunctions
    ✓ ERC1538Query - functionByIndex (324ms)
    ✓ ERC1538Query - functionById (300ms)
    ✓ ERC1538Query - functionExists (262ms)
    ✓ ERC1538Query - functionSignatures (161ms)
    ✓ ERC1538Query - delegateFunctionSignatures (200ms)
    ✓ ERC1538Query - delegateAddress (226ms)
    ✓ ERC1538Query - delegateAddresses (42ms)
    ✓ ERC1538 - receive (81ms)
    ✓ ERC1538 - fallback (75ms)
    ✓ ERC1538 - no update
    ✓ ERC1538 - remove fallback (48ms)

  Contract: ERC1538
    ✓ Ownership (81ms)
    ✓ ERC1538Query - totalFunctions
    ✓ ERC1538Query - functionByIndex (258ms)
    ✓ ERC1538Query - functionById (242ms)
    ✓ ERC1538Query - functionExists (224ms)
    ✓ ERC1538Query - functionSignatures (125ms)
    ✓ ERC1538Query - delegateFunctionSignatures (185ms)
    ✓ ERC1538Query - delegateAddress (227ms)
    ✓ ERC1538Query - delegateAddresses (38ms)
    ✓ ERC1538 - receive (78ms)
    ✓ ERC1538 - fallback (87ms)
    ✓ ERC1538 - no update (46ms)
    ✓ ERC1538 - remove fallback (59ms)

  Contract: GenericFactory
    createContract
      ✓ select random salt
      ✓ predict address
```

```
Started ganache daemon (pid=41839)
Compiling ... success
Migrating ... success
Running tests ... Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

# web3 version: 1.2.1
Chainid is: 1589478046040
Chaintype is: private
Checking factory availability
→ Factory is available on this network
# web3 version: 1.2.1
Chainid is: 1589478046040
Chaintype is: private
Deployer is: 0x5132931eec048e21237A61611E9B0E3f45740A81
[factoryDeployer] IexecLibOrders_v5
[factory] Preparing to deploy IexecLibOrders_v5 ...
[factory] IexecLibOrders_v5 successfully deployed at 0xEb13F139AAc341c7Af
[factoryDeployer] ERC1538UpdateDelegate
[factoryDeployer] ERC1538QueryDelegate
[factoryDeployer] IexecAccessorsDelegate
[factoryDeployer] IexecAccessorsABILegacyDelegate
[factoryDeployer] IexecCategoryManagerDelegate
[factoryDeployer] IexecERC20Delegate
[factoryDeployer] IexecEscrowTokenDelegate
[factoryDeployer] IexecMaintenanceDelegate
[factoryDeployer] IexecOrderManagementDelegate
[factoryDeployer] IexecPocoDelegate
[factoryDeployer] IexecRelayDelegate
[factoryDeployer] ENSIntegrationDelegate
[factoryDeployer] IexecMaintenanceExtraDelegate
[factory] Preparing to deploy ERC1538UpdateDelegate ...
[factory] Preparing to deploy ERC1538QueryDelegate ...
[factory] Preparing to deploy IexecAccessorsDelegate ...
[factory] Preparing to deploy IexecAccessorsABILegacyDelegate ...
[factory] Preparing to deploy IexecCategoryManagerDelegate ...
[factory] Preparing to deploy IexecERC20Delegate ...
[factory] Preparing to deploy IexecEscrowTokenDelegate ...
[factory] Preparing to deploy IexecRelayDelegate ...
[factory] Preparing to deploy ENSIntegrationDelegate ...
[factory] Preparing to deploy IexecMaintenanceExtraDelegate ...
[factory] Preparing to deploy IexecMaintenanceDelegate ...
[factory] Preparing to deploy IexecOrderManagementDelegate ...
[factory] Preparing to deploy IexecPocoDelegate ...
[factory] ERC1538UpdateDelegate successfully deployed at 0x910E52F82235A0
```

Appendix 4 - Disclosure

ConsenSys Diligence ("CD") typically receives compensation from one or more clients (the "Clients") for performing the analysis contained in these reports (the "Reports"). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not [illegible] ould not be interpreted as considering or having any be [illegible] ential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of t[illegible] not a guarantee as to the absolute security of [illegible] to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., "third parties") – on its website. CD hopes that by making these analyses publicly

a
v
a
i
l
a
b
l
e
,
i
t
c
a
n
h
e
l
p
t
h
e
b
l
o
c
k
c
h
a
i
n
e
c
o
s
y

s
t
e
m
d
e
v
e
l
o
p
t
e
c
h
n
i
c
a
l
b
e
s
t
p
r
a
c
t
i
c
e
s
i
n
t
h

i
s
r
a
p
i
d
l
y
e
v
o
l
v
i
n
g
a
r
e
a
o
f
i
n
n
o
v
a
t
i
o
n
.

L
I
N

K
S
T
O
O
T
H
E
R
W
E
B
S
I
T
E
S
F
R
O
M
T
H
I
S
W
E
B
S
I
T
E
Y
o
u
m
a

y
,
t
h
r
o
u
g
h
h
y
p
e
r
t
e
x
t
o
r
o
t
h
e
r
c
o
m
p
u
t
e
r
l
i
n
k

s
,
g
a
i
n
a
c
c
e
s
s
t
o
w
e
b
s
i
t
e
s
o
p
e
r
a
t
e
d
b
y
p
e
r
s
o

n
s
o
t
h
e
r
t
h
a
n
C
o
n
s
e
n
S
y
s
a
n
d
C
D
.
S
u
c
h
h
y
p
e
r
l
i

n
k
s
a
r
e
p
r
o
v
i
d
e
d
f
o
r
y
o
u
r
r
e
f
e
r
e
n
c
e
a
n
d
c
o
n
v

e
n
i
e
n
c
e
o
n
l
y
,
a
n
d
a
r
e
t
h
e
e
x
c
l
u
s
i
v
e
r
e
s
p
o
n
s

i
b
i
l
i
t
y
o
f
s
u
c
h
w
e
b
s
i
t
e
s
’
o
w
n
e
r
s
.
Y
o
u
a
g
r
e
e

t
h
a
t
C
o
n
s
e
n
S
y
s
a
n
d
C
D
a
r
e
n
o
t
r
e
s
p
o
n
s
i
b
l
e
f
o

r
t
h
e
c
o
n
t
e
n
t
o
r
o
p
e
r
a
t
i
o
n
o
f
s
u
c
h
W
e
b
s
i
t
e
s
,

a
n
d
t
h
a
t
C
o
n
s
e
n
S
y
s
a
n
d
C
D
s
h
a
l
l
h
a
v
e
n
o
l
i
a
b
i

l
i
t
y
t
o
y
o
u
o
r
a
n
y
o
t
h
e
r
p
e
r
s
o
n
o
r
e
n
t
i
t
y
f
o
r
t

h
e
u
s
e
o
f
t
h
i
r
d
p
a
r
t
y
W
e
b
s
i
t
e
s
.
E
x
c
e
p
t
a
s
d
e
s

c
r
i
b
e
d
b
e
l
o
w
,
a
h
y
p
e
r
l
i
n
k
f
r
o
m
t
h
i
s
w
e
b
S
i
t
e

t
o
a
n
o
t
h
e
r
w
e
b
s
i
t
e
d
o
e
s
n
o
t
i
m
p
l
y
o
r
m
e
a
n
t
h
a

t
C
o
n
s
e
n
S
y
s
a
n
d
C
D
e
n
d
o
r
s
e
s
t
h
e
c
o
n
t
e
n
t
o
n
t
h

a
t
W
e
b
s
i
t
e
o
r
t
h
e
o
p
e
r
a
t
o
r
o
r
o
p
e
r
a
t
i
o
n
s
o
f
t

h
a
t
s
i
t
e
.
Y
o
u
a
r
e
s
o
l
e
l
y
r
e
s
p
o
n
s
i
b
l
e
f
o
r
d
e
t

e
r
m
i
n
i
n
g
t
h
e
e
x
t
e
n
t
t
o
w
h
i
c
h
y
o
u
m
a
y
u
s
e
a
n
y
c

o
n
t
e
n
t
a
t
a
n
y
o
t
h
e
r
w
e
b
s
i
t
e
s
t
o
w
h
i
c
h
y
o
u
l
i
n

k
f
r
o
m
t
h
e
R
e
p
o
r
t
s
.
C
o
n
s
e
n
S
y
s
a
n
d
C
D
a
s
s
u
m
e
s

n
o
r
e
s
p
o
n
s
i
b
i
l
i
t
y
f
o
r
t
h
e
u
s
e
o
f
t
h
i
r
d
p
a
r
t
y

s
o
f
t
w
a
r
e
o
n
t
h
e
W
e
b
S
i
t
e
a
n
d
s
h
a
l
l
h
a
v
e
n
o
l
i
a

b
i
l
i
t
y
w
h
a
t
s
o
e
v
e
r
t
o
a
n
y
p
e
r
s
o
n
o
r
e
n
t
i
t
y
f
o

r
t
h
e
a
c
c
u
r
a
c
y
o
r
c
o
m
p
l
e
t
e
n
e
s
s
o
f
a
n
y
o
u
t
c
o
m

e
g
e
n
e
r
a
t
e
d
b
y
s
u
c
h
s
o
f
t
w
a
r
e
.

T
I
M
E
L
I
N
E
S
S
O
F

CONTENTTheecontentcontainedintheReport

s
i
s
c
u
r
r
e
n
t
a
s
o
f
t
h
e
d
a
t
e
a
p
p
e
a
r
i
n
g
o
n
t
h
e
R
e

p
o
r
t
a
n
d
i
s
s
u
b
j
e
c
t
t
o
c
h
a
n
g
e
w
i
t
h
o
u
t
n
o
t
i
c
e

.
U
n
l
e
s
s
i
n
d
i
c
a
t
e
d
o
t
h
e
r
w
i
s
e
,
b
y
C
o
n
s
e
n
S
y
s

a
n
d
C
D
.