Celer Network

Security Assessment

October 23rd, 2020

For :

Celer Network

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Celer Network
Description	The code-base comprise DPoS contract, State Guardian Network (SGN) contract, Example ERC-20 token implementation and library contracts to decode `protobuf` encoded data. DPoS contract act as a single point to manage all operations of validators, delegators and sidechain contracts such as SGN.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commit	4e8ef997047c17e35c6194716d423709fe6e8371

Audit Summary

Delivery Date	Oct 23, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Oct. 05, 2020 - Oct. 10 2020

Vulnerability Summary

Total Issues	49
Total Critical	0
Total Major	0
Total Minor	3
Total Informational	46

Executive Summary

The codebase comprise of contracts implementing logic for operations related to `Celer` sidechain. The contracts make use of `OpenZeppelin` contracts to implement `ERC20` token, pausable, whitelisting and ownable functionalities.
All of the contracts in repository were reviewed and majority of the findings are `informational` for enhancing the optimization and code legibility of the contracts.

Findings

ID	Title	Type	Severity
GEX-01	Unlocked Compiler Version	Compiler Version	Informational
GOV-01	Inefficient usage of `SafeMath`	Optimization	Informational
GOV-02	Missing check for the valid input	Code Legibility	Informational
GOV-03	Substitution of require calls with Modifier	Coding Style	Informational
GOV-04	Substitution of require calls with Modifier	Coding Style	Informational
GOV-05	Substitution of require calls with Modifier	Coding Style	Informational
GOV-06	Incorrect order of functions	Language Specific	Informational
GOV-07	`Storage` Updated after External Call	External Interaction	Informational
GOV-08	`Storage` Updated after External Call	External Interaction	Informational
GOV-09	Function visibility can be changed to `external`	Language Specific	Informational
SGN-01	Ineffectual Library import and declaration	Coding Style	Informational
SGN-02	Variable not named in `camel case`	Coding Style	Informational
SGN-03	Spelling Error	Comment	Informational
SGN-04	Potentially Incorrect Error Message	Coding Style	Informational
SGN-05	Incorrect Comparison	Implementation	Minor
SGN-06	Ability of Owner to withdraw at Will	Code Legibility	Informational

ID	Title	Type	Severity
PBU-01	Usage of `uint` alias instead of `uint256`	Coding Style	Informational
PBU-02	Redundant initialization with default value	Optimization	Informational
PBU-03	Potential Overflow of bytes Array	Array Overflow	Minor
PBU-04	Inexistence of `reason` in Require Statements	Coding Style	Informational
PGN-01	Usage of `uint` alias instead of `uint256`	Coding Style	Informational
PGN-02	Ineffectual `if` statement	Optimization	Informational
DPS-01	`TODO` comment	Comment	Informational
DPS-02	Spelling Error	Comment	Informational
DPS-03	Redundant State Variable	Optimization	Informational
DPS-04	Spelling Error	Comment	Informational
DPS-05	Ability of Owner to withdraw at Will	Code Legibility	Informational
DPS-06	Inefficient use of local variable	Optimization	Informational
DPS-07	Inefficient use of local variable	Optimization	Informational
DPS-08	Substitution of require calls with Modifier	Coding Style	Informational
DPS-09	Redundant initialization with default value	Optimization	Informational
DPS-10	Redundant initialization with default value	Optimization	Informational

ID	Title	Type	Severity
DPS-11	Confusing Variable Name	Code Legibility	Informational
DPS-12	`require` statement can be substituted with a function	Coding Style	Informational
DPS-13	`require` statement can be substituted with a function	Code Legibility	Informational
DPS-14	Inexistence of `reason` in Require Statements	Coding Style	Informational
DPS-15	Missing check for the valid input	Coding Style	Informational
DPS-16	Usage of `ether` instead of Decimals Multiplier	Code Legibility	Informational
DPS-17	Inefficient Code	Optimization	Informational
DPS-18	Potentially Incorrect Comparison	Code Legibility	Informational
DPS-19	Potentially Incorrect Implementation	Code Legibility	Minor
DPS-20	Usage of `storage` Variable Instead of `memory`	Optimization	Informational
DPS-21	Inefficient `storage` Read	Optimization	Informational
DPS-22	Inefficient Code	Optimization	Informational
DPS-23	Inefficient use of local variable	Optimization	Informational
DPS-24	Function visibility can be changed to `external`	Language Specific	Informational
CER-01	Can be declared `constant` and use `uint256` instead of alias `uint`	Optimization	Informational
DPC-01	Spelling Error	Comment	Informational
DPC-02	Grammar Error	Comment	Informational

GEX-01: Unlocked Compiler Version

Type	Severity	Location
Compiler Version	Informational	All Contracts

Description:

The contracts have unlocked compiler versions. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation:

We advise that the compiler version is instead locked at the lowest version possible that the contracts can be compiled at f.e. the contracts can be safely locked at v0.5.0.

```
pragma solidity 0.5.0;
```

Alleviation:

Alleviations were applied as advised

GOV-01: Inefficient usage of `SafeMath`

Type	Severity	Location
Optimization	Informational	Govern.sol L137, L151, L219, L233

Description:

The Addition and Subtraction operations on the aforementioned lines do not depend upon the values received outside of the contract and hence their chances of `oveflowing` or `underflowing` are neglibible considering a `uint256` can store a large enough value for their use cases. These operations can be safely performed without `SafeMath` which will save the additional gas cost associated with Safe Addition and Safe Subtraction without comprising the security of the contract.

Recommendation:

We advise that the `SafeMath` library usage on the aforementioned lines be replaced with UnSafe Addition and Subtraction.

We advise following changes for the code.

```
//L137 nextParamProposalId = nextParamProposalId.add(1);
// TO
nextParamProposalId = nextParamProposalId + 1;
```

```
//L151 nextParamProposalId.sub(1),
// TO
nextParamProposalId - 1,
```

```
//L219 nextSidechainProposalId = nextSidechainProposalId.add(1);
// TO
nextSidechainProposalId = nextSidechainProposalId + 1;
```

```
//L233 nextSidechainProposalId.sub(1),
// TO
nextSidechainProposalId - 1,
```

Alleviation:

Alleviations were applied as advised.

GOV-02: Missing check for the valid input

Type	Severity	Location
Code Legibility	Informational	Govern.sol L167, L249

Description:

The functions on the aforementioned lines has parameter `_vote` of type `VoteType`. The code does not have a check against a `VoteType` value of `VoteType.UnVoted`, which if passed would successfully execute the transaction without changing the state of the vote.

Recommendation:

We recommend that a `require` statement is added in the functions on aforementioned lines to check against `VoteType.Unvoted` value of parameter `_vote`.

We advise following changes for the code.

```
require(
    _vote != VoteType.UnVoted,
    "_vote cannot be Unvoted"
);
```

Alleviation:

Client suggested that Vote is allowed to be overwritten to Unvoted state by the voter and hence the exhibit was not applicable.

GOV-03: Substitution of require calls with Modifier

Type	Severity	Location
Coding Style	Informational	Govern.sol L173, L190, L255, L272

Description:

The `require` statement on the aforementioned lines can be converted into a modifier to avoid code duplication and increase the legibility of the code.

Recommendation:

We recommend that a `require` statement on the aforementioned lines be converted into a modifier and used instead, in their respective function signatures.

We advise following changes for the code.

```
modifier isVoting(uint256 _proposalId) {
    require(
        paramProposals[_proposalId].status == ProposalStatus.Voting,
        'Invalid proposal status'
    );
    _;
}
// Usage
function () internal isVoting(_proposalId) {...}
```

Alleviation:

Client chose not to apply alleviation stating that proposals are different and duplication is not significant enough.

GOV-04: Substitution of require calls with Modifier

Type	Severity	Location
Coding Style	Informational	Govern.sol L256, L174

Description:

The `require` statement on the aforementioned lines can be converted into a modifier to avoid code duplication and increase the legibility of the code.

Recommendation:

We recommend that a `require` statement on the aforementioned lines be converted into a modifier and used instead, in their respective function signatures.

We advise following changes for the code.

```
modifier deadlineNotReached(uint256 _proposalId) {
    require(
        block.number < paramProposals[_proposalId].voteDeadline,
        'Vote deadline reached'
    );
    _;
}
// Usage
function () internal deadlineNotReached(_proposalId) {...}
```

Alleviation:

Client chose not to apply alleviation stating that proposals are different and duplication is not significant enough.

GOV-05: Substitution of require calls with Modifier

Type	Severity	Location
Coding Style	Informational	Govern.sol L175, L257

Description:

The `require` statement on the aforementioned lines can be converted into a modifier to avoid code duplication and increase the legibility of the code.

Recommendation:

We recommend that a `require` statement on the aforementioned lines be converted into a modifier and used instead, in their respective function signatures.

We advise following changes for the code.

```
modifier alreadyNotVoted(uint256 _proposalId) {
    require(
        paramProposals[_proposalId].votes[_voter] == VoteType.Unvoted,
        'Voter has voted'
    );
    _;
}
// Usage
function () internal alreadyNotVoted(_proposalId) {...}
```

Alleviation:

Client chose not to apply alleviation stating that proposals are different and duplication is not significant enough.

GOV-06: Incorrect order of functions

Type	Severity	Location
Language Specific	Informational	Govern.sol

Description:

The structure of the codebase does not conform to the official Solidity style guide of `v0.5.x`.

Recommendation:

An indicative excerpt of the style guide is that functions should be grouped according to their visibility and ordered:

```
constructor
fallback function (if exists)
external
public
internal
private
```

Within a grouping, place the `view` and `pure` functions last.

Alleviation:

No alleviations.

GOV-07: `Storage` Updated after External Call

Type	Severity	Location
External Interaction	Informational	Govern.sol L195-L196

Description:

The code on the aforementioned lines update `storage` after making an external call which violates the `checks-effects-interactions` pattern and poses threat for `reentrancy` attack.

Reference the Check Effects Interactions pattern:
https://fravoll.github.io/solidity-patterns/checks_effects_interactions.html

Recommendation:

We advise that the code be rectified to make `storage` changes first before making an external.

We advise following changes for the code.

```
if (_passed) {
    UIntStorage[p.record] = p.newValue;
    governToken.safeTransfer(p.proposer, p.deposit);
}
```

Alleviation:

No alleviations.

GOV-08: `Storage` Updated after External Call

Type	Severity	Location
External Interaction	Informational	Govern.sol L277-L278

Description:

The code on the aforementioned lines update `storage` after making an external call which violates the `checks-effects-interactions` pattern and poses threat for `reentrancy` attack.

Reference the Check Effects Interactions pattern:
https://fravoll.github.io/solidity-patterns/checks_effects_interactions.html

Recommendation:

We advise that the code be rectified to make `storage` changes first before making an external.

We advise following changes for the code.

```
if (_passed) {
    registeredSidechains[p.sidechainAddr] = p.registered;
    governToken.safeTransfer(p.proposer, p.deposit);
}
```

Alleviation:

No alleviations.

GOV-09: Function vibility can be changed to `external`

Type	Severity	Location
Language Specific	Informational	Govern.sol L135, L217

Description:

The functions on the aforementioned lines are never called from within the contract and hence their visibilities can be changed to `external`.

Recommendation:

We recommend to change the visibilites of function on the aforementioned lines to `external`.

Alleviation:

Alleviations were applied as advised.

SGN-01: Ineffectual Library import and declaration

Type	Severity	Location
Coding Style	Informational	SGN.sol L6, L22

Description:

The library `ECDSA` is imported and declared but is never used in the contract. The import and declaration on the aforementioned lines can be safely removed.

Recommendation:

We advise that the import and declaration of library `ECDSA` on the aforementioned lines be removed.

Alleviation:

Alleviations were applied as advised.

SGN-02: Variable not named in `camelCase`

Type	Severity	Location
Coding Style	Informational	SGN.sol L25

Description:

The state variable `DPoSContract` of type `IDPoS` is not named in `camelCase` which violates the best coding practices of Solidity.

Recommendation:

We advise that the variable `DPoSContract` be renamed to `dPoSContract` and used in all of the occurences in the contract.

Alleviation:

Alleviations were applied as advised.

SGN-03: Spelling Error

Type	Severity	Location
Comment	Informational	SGN.sol L52

Description:

The comment on the aforementioned line has a spelling error in the word `Owner`.

Recommendation:

We advise that the spellings error is corrected on the aforementioned line.

Alleviation:

Alleviations were applied as advised.

SGN-04: Potentially Incorrect Error Message

Type	Severity	Location
Coding Style	Informational	SGN.sol L106

Description:

The error message on the aforementioned line has a potentially incorrect error message `Fail to check validator sigs` which implies that the operation of validating signatures itself failed to initiate. The `require` statement throws when the signatures are invalid and as such the error message should highlight it.

Recommendation:

We advise that the error message be changed to `validator sigs verification failed`.

Alleviation:

Alleviations were applied as advised.

SGN-05: Incorrect Comparison

Type	Severity	Location
Implementation	Minor	SGN.sol L115

Description:

The `require` statement on the aforementioned line checks against that the `servicePool` should always be greater than `newServiceReward` which will result in not being able to claim all of funds specified by `servicePool`.

Recommendation:

We advise that the conditional in the `require` statement be changed to `greater-than-or-equal` so that all the funds in `servicePool` are claimable.

We advise following changes for the code.

```
require(
    servicePool >= newServiceReward,
    'Service pool is smaller than new service reward'
);
```

Alleviation:

Alleviations were applied as advised.

SGN-06: Ability of Owner to withdraw at Will

Type	Severity	Location
Code Legibility	Informational	SGN.sol L56

Description:

The function on the aforementioned line allows the `owner` of contract to withdraw `celerToken` funds at will. Although, it does not compromise the security of the system in any way but the `owner` has to be trusted to maintain the funds of the contract.

Recommendation:

There are no recommendations for this exhibit.

Alleviation:

No alleviations were needed.

PBU-01: Usage of `uint` alias instead of `uint256`

Type	Severity	Location
Coding Style	Informational	Pb.sol

Description:

The library is using `uint` to declare 256-bit unsigned integers. Although, `uint` is an alias for `uint256` and both represent the same underlying integer allocation. It is advisable that for clean coding practices the complete form `uint256` should be used instead of the alias `uint`.

Recommendation:

We advise to use `uint256` instead of alias `uint` in all of the occurrences in the contract.

Alleviation:

Alleviations were optional and client chose not to apply the alleviations as the code was auto-generated.

PBU-02: Redundant initialization with default value

Type	Severity	Location
Optimization	Informational	Pb.sol L15, L98

Description:

The aforementioned lines declare variables of type `uint256` and initialize it with `0`. In Solidity, all un-initialized variables have a default value which for the `uint256` variable is `0`, hence the initialization part is redundant and can be removed.

Recommendation:

We recommend that the explicit initialization of type `uin256` with default value `0` be removed as it is redundant.

Alleviation:

Alleviations were partly applied as alleviation was not applied on `L15`.

PBU-03: Potential Overflow of `bytes` Array

Type	Severity	Location
Array Overflow	Minor	Pb.sol L72-L73, L93-L94, L115-L116

Description:

The addition of `idx` and `len` is `Unsafe` and can potentially overflow resulting in the followed `require` check potentially passing despite the overflow.
Additionally, The `require` statements on the aforementioned lines check against that the current read index `idx` should not exceed the length of `buf`, which is of type `bytes`. As the `idx` starts from `0` and the length of `buf` starts from `1`, if in any case `idx` is equal to length of `buf`, the `require` statement evaluates to `true` and the memory read in that case would not be a part of `buf` as the `idx` would have oveflown.

Recommendation:

We advise that the `SafeMath` library be used for the `Safe` addition of `idx` and `len` and the conditional in `require` statements changed from `less-than-or-equal-to` to `less-than`.

We advise following changes for the code.

```
uint256 end = buf.idx.add(len);
require(end < buf.b.length) // end is `idx`
```

Alleviation:

Client suggested that vector for overflow is negligible as the `end` is not used for current read, but points to the next read start index. There is also a hasMore() function to decide whether a next read should happen. No alleviations.

PBU-04: Inexistence of `reason` in Require Statements

Type	Severity	Location
Coding Style	Informational	Pb.sol L73, L94, L116, L126, L136, L142

Description:

The `require` statements on the aforementioned lines do not specify `reason string`. The `reason string`s should be provided to `require` statements to aid debugging and readability of the code.

Recommendation:

We advise that the `reason string`s be added to the `require` statements on the aforementioned lines.

Alleviation:

Alleviations were optional and client chose not to apply the alleviations as the code was auto-generated.

PGN-01: Usage of `uint` alias instead of `uint256`

Type	Severity	Location
Coding Style	Informational	PbSgn.sol

Description:

The library is using `uint` to declare 256-bit unsigned integers. Although, `uint` is an alias for `uint256` and both represent the same underlying integer allocation. It is advisable that for clean coding practices the complete form `uint256` should be used instead of the alias `uint`. As the comment at the top of library indicates that it is a code generated library and the rectification may require changes in the tool used to generate this library. The exhibit is `informational` and can be safely ignored.

Recommendation:

We advise to use `uint256` instead of alias `uint` in all of the occurrences in the contract.

Alleviation:

Alleviations were optional and client chose not to apply the alleviations as the code was auto-generated.

PGN-02: Ineffectual `if` statement

Type	Severity	Location
Optimization	Informational	PbSgn.sol L25, L114, L148, L172

Description:

The `if` statements on the aforementioned lines have hardcoded `false` conditional as predicate which makes the `if` statement and its block ineffectual. As the comment at the top of library indicates that it is a code generated library and the rectification may require changes in the tool used to generate this library. The exhibit is `informational` and can be safely ignored.

Recommendation:

We recommend to remove the `if` clause from the aforementioned lines and converting the following `else-if` clause to `if` clause.

Alleviation:

Alleviations were optional and client chose not to apply the alleviations as the code was auto-generated.

DPS-01: `TODO` comment

Type	Severity	Location
Comment	Informational	DPoS.sol L68

Description:

The aforementioned line has a `TODO` which we advise that be removed from code.

Recommendation:

We advise the removal of `TODO` comment on the aforementioned line.

Alleviation:

Alleviations were applied as advised.

DPS-02: Spelling Error

Type	Severity	Location
Comment	Informational	DPoS.sol L142

Description:

The aforementioned line has a spelling error for the word `creation`.

Recommendation:

We advise that the spelling of `creation` be corrected on the aforementioned line.

Alleviation:

Alleviations were applied as advised.

DPS-03: Redundant State Variable

Type	Severity	Location
Optimization	Informational	DPoS.sol L167

Description:

The variable `celerToken` on the aforementioned line is initialized with the same address as `governToken` variable in the parent `Govern` contract on `L71`. As both the variables are initialized with the same `ERC-20` token address, one of them can removed to save gas costs associated with additional storage slot and to avoid the duplicate code to achieve code legibility.

Recommendation:

We advise that the state variable `celerToken` is removed from the `DPoS` contract and instead `governToken` variable inherited from `Govern` contract be used in all of the occurrences of `celerToken` in the `DPoS` contract.

Alleviation:

Alleviations were applied as advised.

DPS-04: Spelling Error

Type	Severity	Location
Comment	Informational	DPoS.sol L189

Description:

The comment on the aforementioned line has spelling error for the word `Owner`.

Recommendation:

We advise that the spelling of word `Owner` are corrected on the aforementioned line.

Alleviation:

Alleviations were applied as advised.

DPS-05: Ability of Owner to withdraw at Will

Type	Severity	Location
Code Legibility	Informational	DPoS.sol L189

Description:

The function on the aforementioned line allows the `owner` of contract to withdraw `celerToken` funds at will. Although, it does not compromise the security of the system in any way but the `owner` has to be trusted to maintain the funds of the contract.

Recommendation:

There are no recommendations for this exhibit.

Alleviation:

No alleviations were needed for this exhibit.

DPS-06: Inefficient use of local variable

Type	Severity	Location
Optimization	Informational	DPoS.sol L214

Description:

The declaration of local variable on the aforementioned line is inefficient as the variable is never used at more than one places. The gas cost associated with local variable declaration can be saved by using the initialization part directly in place of the declared local variable.

Recommendation:

We advise that the initialization part of the local variable declaration be directly used in place of the local variable.

We advise following changes for the code.

```
// Usage
for (uint256 i = 0; i < getUIntValue(uint256(ParamNames.MaxValidatorNum)); i++) {...}
```

Alleviation:

The exhibit was ignored as it was incorrectly identified.

DPS-07: Inefficient use of local variable

Type	Severity	Location
Optimization	Informational	DPoS.sol L248

Description:

The declaration of local variable on the aforementioned line is inefficient as the variable is never used at more than one places. The gas cost associated with local variable declaration can be saved by using the initialization part directly in place of the declared local variable.

Recommendation:

We advise that the initialization part of the local variable declaration be directly used in place of the local variable.

We advise following changes for the code.

```
// Usage
for (uint256 i = 0; i < getUIntValue(uint256(ParamNames.MaxValidatorNum)); i++) {...}
```

Alleviation:

The exhibit was ignored as it was incorrectly identified.

DPS-08: Substitution of require calls with Modifier

Type	Severity	Location
Coding Style	Informational	DPoS.sol L204, L238

Description:

The `require` statements on the aforementioned lines can be converted to a modifier to avoid code duplication and increase legibility of the code.

Recommendation:

We advise that the `require` statements on the aforementioned lines be converted to a modifier.

We advise following changes for the code.

```
modifier onlyValidator() {
    require(
        isValidator(msgSender),
        'msg sender is not a validator'
    );
    _;
}
// Usage
function voteParam(uint256 _proposalId, VoteType _vote) external onlyValidator {...}
function voteSidechain(uint256 _proposalId, VoteType _vote) external onlyValidator {...}
```

Alleviation:

Alleviations were applied as advised.

DPS-09: Redundant initialization with default value

Type	Severity	Location
Optimization	Informational	DPoS.sol L217, L251, L424, L537, L543, L577, L610, L678, L790, L926

Description:

The aforementioned lines declare variables of type `uint256` and initialize it with `0`. In Solidity, all un-initialized variables have a default value which for the `uint256` variable is `0`, hence the initialization part is redundant and can be removed.

Recommendation:

We recommend that the explicit initialization of type `uin256` with default value `0` be removed as it is redundant.

Alleviation:

Alleviations were applied as advised.

DPS-10: Redundant initialization with default value

Type	Severity	Location
Optimization	Informational	DPoS.sol L927

Description:

The aforementioned line declares variablesof type `bool` and initialize it with `false`. In Solidity, all un-initialized variables have a default value which for the `bool` variable is `false`, hence the initialization part is redundant and can be removed.

Recommendation:

We recommend that the explicit initialization of type `bool` with default value `false` be removed as it is redundant.

Alleviation:

Alleviations were applied as advised.

DPS-11: Confusing Variable Name

Type	Severity	Location
Code Legibility	Informational	DPoS.sol L217, L251

Description:

The variable name `yesVotes` on the aforementioned line implies that it represents the count of `yes` votes yet it specifies the amount of total stakes represented by validators with `yes` votes. The variable name should reflect the staking amount it represents.

Recommendation:

We recommend that the variable be renamed to f.e. `yesVotesStakeAmt` and be used in all of its ocurrences.

Alleviation:

Alleviations were applied as advised.

DPS-12: `require` statement can be substituted with a modifier

Type	Severity	Location
Coding Style	Informational	DPoS.sol L329, L342, L357, L373, L395, L411

Description:

The `require` statements on the aforementioned lines have the same predicates and error message. The require statement can be converted into a modifier to avoid code duplication and increase legibility of the code.

Recommendation:

We advise that the `require` statement on the aforementioned lines be converted into a modifier for assertion.

We advise following changes for the code.

```
modifier isCandidateInitialized() {
    require(
        candidateProfiles[msg.sender].initialized,
        'Candidate is not initialized'
    );
}
```

Alleviation:

Alleviations were applied as advised.

DPS-13: `require` statement can be substituted with a function

Type	Severity	Location
Code Legibility	Informational	DPoS.sol L330

Description:

The `require` statement on the aforementioned line asserts that the new commission rate should be less-than-or-equal to the already set commission rate. However, the function name `nonIncreaseCommissionRate` implies that the new rate should be less-than the already set rate. Also, the execution of the function will be ineffectual and non-state changing if the provided rate and time in the parameters is same as the already set values.

Recommendation:

We advise that the conditional of `require` statement on the aforementioned line be changed from less-than-or-equal to less-than.

We advise following changes for the code.

```
require(
    _newRate < candidate.commissionRate,
    'Invalid new rate'
);
```

With the application of above code changes, the following change can also applied on `L814` to optimize the code.

```
if (_newRate < _candidate.commissionRate) {...}
```

Alleviation:

Client suggested that `New Rate` can be equal to `Current Rate`, as the operation may be just increasing rate lock end time. Alleviations were not needed.

DPS-14: Inexistence of `reason` in Require Statements

Type	Severity	Location
Coding Style	Informational	DPoS.sol L98, L310, L412, L450, L421, L869

Description:

The `require` statements on the aforementioned lines do not specify `reason string`. The `reason string`s should be provided to `require` statements to aid debugging and readability of the code.

Recommendation:

We advise that the `reason string`s be added to the `require` statements on the aforementioned lines.

Alleviation:

Alleviations were applied as advised.

DPS-15: Missing check for the valid input

Type	Severity	Location
Coding Style	Informational	DPoS.sol L303, L340

Description:

The functions on the aforementioned lines has parameter `lock end time`. There are no checks in the function to assert against a valid value of it which should be greater than `block.number`.

Recommendation:

We advise that a `require` statement or perhaps a modifier is added in the aforementioned functions to assert that the `new lock end` time is greater-than `block.number`.

We advise following addition to the code.

```
require(
    rate > block.number,
    'rate must be greater than block.number'
);
```

Alleviation:

Client suggested that it is the validator's responsibility to provide a valid `lock end time` value so no alleviations were applied.

DPS-16: Usage of `ether` instead of Decimals Multiplier

Type	Severity	Location
Code Legibility	Informational	DPoS.sol L392, L467, L491

Description:

The `modifier` usage on the aforementioned passes minimum of token as `1 CELR`. The decimals multiplier used is `ether` global variable which represents 18 decimals. Although, it is functionally correct but we advise that actual decimals multiplier of the token be used instead of `ether` global variable for better readability of the code.

Recommendation:

We advise that the actual decimals multiplier of the token be used in place of the `ether` global variable.

We advise following changes to the code.

```
uint256 constant DECIMALS_MULTIPLIER = 10**18;
// Usage
minAmount(_amount, 1 * DECIMALS_MULTIPLIER)
```

Alleviation:

Alleviations were applied as advised.

DPS-17: Inefficient Code

Type	Severity	Location
Optimization	Informational	DPoS.sol L823-L825

Description:

The code on the aforementioned lines only effectually executes when the function `_updateCommissionRate` is called from `confirmIncreaseCommissionRate` as part of `announced` rate update. When the function `_updateCommissionRate` is called from `nonIncreaseCommissionRate`, the execution of aforementioned lines is ineffectual and hence costs unwanted gas. The aforementioned lines can be moved to `confirmIncreaseCommissionRate` after the call to `_updateCommissionRate` so that the execution of `nonIncreaseCommissionRate` does not execute the code on aforementioned lines.

Recommendation:

We advise that the code from aforementioned lines be move to the body of function `confirmIncreaseCommissionRate` after it makes call to `_updateCommissionRate`.

We advise following changes to the code.

```
function confirmIncreaseCommissionRate() external {
        ValidatorCandidate storage candidate = candidateProfiles[msg.sender];
        require(candidate.initialized, 'Candidate is not initialized');
        uint256 advanceNoticePeriod = getUIntValue(uint256(ParamNames.AdvanceNoticePeriod));
        require(
            block.number > candidate.announcementTime.add(advanceNoticePeriod),
            'Still in notice period'
        );
        _updateCommissionRate(candidate, candidate.announcedRate,
candidate.announcedLockEndTime);
        delete _candidate.announcedRate;
        delete _candidate.announcedLockEndTime;
        delete _candidate.announcementTime;
}
```

Alleviation:

Alleviations were applied as advised.

DPS-18: Potentially Incorrect Comparison

Type	Severity	Location
Code Legibility	Informational	DPoS.sol L416

Description:

The `require` statement on the aforementioned line asserts that `block.number` should be greater than the `earliestBondTime` for the earliest bond time to arrive. However, intiuitively when the `block.number` is equal to `earliestBondTime` then it should be considered that the bond time has arrived.

Recommendation:

We advise that the conditional in `require` statement be changed from greater-than to greater-than-or-equal to take account the arrival of bond time if the `block.Number` and `earliestBondTime` are equal.

We advise following changes to the code.

```
require(
    block.number >= candidate.earliestBondTime,
    'Not earliest bond time yet'
);
```

Alleviation:

Alleviations were applied as advised.

DPS-19: Potentially Incorrect Implementation

Type	Severity	Location
Code Legibility	Minor	DPoS.sol L247 - L260

Description:

The function `confirmSidechainProposal` does not handle the assignment of deposited funds when a sidechain proposal does not pass. In the function `confirmParamProposal`, when a Param proposal does not pass, the deposited funds are assigned to `miningPool` but no such assignment of funds is implemented in `confirmSidechainProposal`.

Recommendation:

We advise that the code is added to handle the assignment of funds in case when a sidechain proposal does not pass. If the current implementation is intentional then this exhibit can be safely ignored.

Alleviation:

Alleviations were applied as advised.

DPS-20: Usage of `storage` Variable Instead of `memory`

Type	Severity	Location
Optimization	Informational	DPoS.sol L731, L761

Description:

The local variables on the aforementioned lines are declared as `storage` yet no write operations are performed. The usage of local `storage` variables is optimal in read operations only when the count of read operations is less than 4. As both of the aforementioned local `storage` variables perform more than 3 read operations, their data location can be changed to `memory` as it costs significantly less gas to read from `memory`. Although, the usage of `memory` data location will copy the data to memory but it is still cheaper than to make several read operations on `storage`.

Recommendation:

We advise that the data location of the variables on the aforementioned lines be changed from `storage` to `memory`.

Alleviation:

Alleviations were applied as advised.

DPS-21: Inefficient `storage` Read

Type	Severity	Location
Optimization	Informational	DPoS.sol L766 - L769

Description:

The `for-loop` on the aforementioned lines makes use of inefficient `storage` read operations on the lines `L767` and `L768` with the use of `d.intentStartIndex`. Although, the enclosing function is a `view` type and is never called from within the contract but a possible call from another contract will be gas inefficient.

Recommendation:

We advise that the code on the aforementioned lines be rectified to minimize the number of `storage` read operations.

We advise following changes for the code.

```
for (uint256 i = d.intentStartIndex; i < len; i++) {
            intentAmounts[i] = d.withdrawIntents[i].amount;
            intentProposedTimes[i] = d.withdrawIntents[i].proposedTime;
}
```

Alleviation:

Original code suggestion turned out to be incorrect with the follow up of correct code suggestion but client chose not apply alleviations stating it is inside an external view function called by the off-chain clients, and we expect a delegator to only have 0 or 1 withdrawal intent in normal cases. So I think gas consumption doesn't matter here and we can keep the code as it is.

DPS-22: Inefficient Code

Type	Severity	Location
Optimization	Informational	DPoS.sol L587 - L606

Description:

The code on the aforementioned lines can be optimized by the use of a local variable and a single `_updateDelegatedStake` call to reduce the `bytecode` footprint of the contract.

Recommendation:

We advise following changes for the code.

```
Delegator storage delegator = validator.delegatorProfiles[penalizedDelegator.account];
uint256 _amt;
if (delegator.delegatedStake >= penalizedDelegator.amt) {
    _amt = penalizedDelegator.amt;
} else {
    uint256 remainingAmt = penalizedDelegator.amt.sub(delegator.delegatedStake);
    delegator.undelegatingStake = delegator.undelegatingStake.sub(remainingAmt);
    _amt = delegator.delegatedStake;
}
_updateDelegatedStake(
    validator,
    penalty.validatorAddress,
    penalizedDelegator.account,
     _amt,
    MathOperation.Sub
);
```

Alleviation:

Alleviations were applied as advised.

DPS-23: Inefficient use of local variable

Type	Severity	Location
Optimization	Informational	DPoS.sol L358, L376, L417, L427, L889, L526, L539, L649, L676, L692, L788, L907, L955

Description:

The declaration of local variables on the aforementioned line is inefficient as any variable is never used at more than one place. The gas cost associated with local variable declaration can be saved by using the initialization part directly in place of the declared local variable.

Recommendation:

We advise that the initialization part of the local variable declaration be directly used in place of the local variable.

Alleviation:

Alleviations were partly applied as some of the suggestion were incorrectly indentified.

DPS-24: Function vibility can be changed to `external`

Type	Severity	Location
Language Specific	Informational	DPoS.sol L691, L718, L751

Description:

The functions on the aforementioned lines are never called from within the contract and hence their visibilities can be changed to `external`.

Recommendation:

We recommend to change the visibilites of function on the aforementioned lines to `external`.

Alleviation:

Alleviations were applied as advised.

CER-01: Can be declared `constant` and use `uint256` instead of alias `uint`

Type	Severity	Location
Optimization	Informational	CELRToken.sol L15

Description:

The variable declaration on the aforementioned line be changed to a constant to save gas cost associate with `storage` slot. Additionaly, the type of declaration is `uint` which is an alias `uint256`. A clean coding practice is to use complete type name of `uint256` instead of alias `uint`.

Recommendation:

We advise that the variable declaration be chanaged to `constant` and complete type name of `uint256` be used.

We advise following changes for the code.

```
uint256 constant public INITIAL_SUPPLY = 1e28;
```

Alleviation:

Alleviations were applied as advised.

DPC-01: Spelling Error

Type	Severity	Location
Comment	Informational	DPoSCommon.sol L12

Description:

The aforementioned line has a spelling error for `misbehaviour`

Recommendation:

We advise that the spelling error for `misbehaviour` be corrected on the aforementioned line.

Alleviation:

Alleviations were applied as advised.

DPC-02: Grammar Error

Type	Severity	Location
Comment	Informational	DPoSCommon.sol L10

Description:

The aforementioned line has a grammar error where it says `Delegators has to wait`.

Recommendation:

We advise that the grammar error be rectified and the comment part changed to `Delegators have to wait`.

Alleviation:

Alleviations were applied as advised.