CERTIK

Security Assessment

# cBridge v2

Nov 17th, 2021

# Table of Contents

# Summary

This report has been prepared for Celer to discover issues and vulnerabilities in the source code of the cBridge v2 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | cBridge v2 |
| Description | State Guardian Network from Celer Network |
| Platform | CosmosSDK, Celer Network |
| Language | Solidity |
| Codebase | https://github.com/celer-network/sgn-v2-contracts/tree/803e5377cfe6142c049a6f5067825272a7e73b6f |
| Commit | 803e5377cfe6142c049a6f5067825272a7e73b6f<br>bad4774b43c5e8abee81b9399fb3fca75d22e61e<br>eaae4e2d705cbbe5025c9e7b1ec9040b126122a6<br>a436e2e1d88d320c4bcfe831d23ac325a214e003<br>5e9cf3159f86c4d2a8695742e5051374cf10bf58<br>c5766f5d10594661198491d3235ea22c6dffc327 |

## Audit Summary

| | |
|---|---|
| Delivery Date | Nov 17, 2021 |
| Audit Methodology | Manual Review, Static Analysis |
| Key Components | |

## Vulnerability Summary

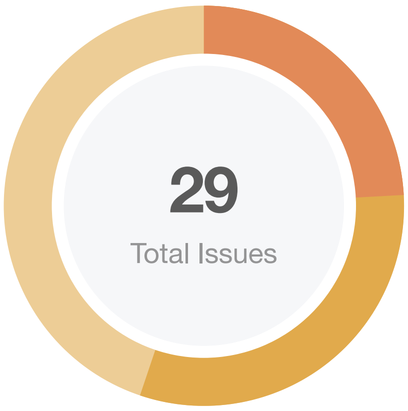
| Vulnerability Level | Total | Pending | Declined | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|---|
| ● Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Major | 7 | 0 | 0 | 7 | 0 | 0 |
| ● Medium | 9 | 0 | 0 | 4 | 0 | 5 |
| ● Minor | 13 | 0 | 0 | 2 | 0 | 11 |
| ● Informational | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit Scope

| ID | File | SHA256 Checksum |
|---|---|---|
| ISV | interfaces/ISigsVerifier.sol | c305e9f3f4b1179005494f7ee4262e8ac5532fd0d70c17c5b0e7926d017ea1e8 |
| REA | libraries/proto/README.md | 9b872eddde04db2b1cc671530beadcc7c2e465756efe4b4df621df66f6292aed |
| BRI | libraries/proto/bridge.proto | 848501ecd3a2331bea320906162c6d3f5d7ffeb1a889e95ed62ec6e2d23652a0 |
| FAR | libraries/proto/farming.proto | ff30bf43bfaaaea9aba23e97c423e2178fb0329c3af1b19d96b8cd73d3578e7a |
| POO | libraries/proto/pool.proto | 0188e88bbf2a89c0ace97cf73472b2716b50a8b1c5739ce965f27407f4e25b25 |
| SGN | libraries/proto/sgn.proto | 4af43f06f23a5756a8f7983d7aa0882be46e23166962a7794ab1934c61a21fa3 |
| STA | libraries/proto/staking.proto | 3f91795ee52894f10dcbd0f9e49c832e2723f48da5738369454a0d29e0d14510 |
| DTC | libraries/DataTypes.sol | c3d63e448008855a9eaff5c48eea02dd39e1c9a924dc5397cdfb233ae742791e |
| PCK | libraries/Pb.sol | 7d86a7759dfaf99794b7cd50202151688e4d25682bf99a8ecc04fc5a18ffd59b |
| PBC | libraries/PbBridge.sol | 723d3762e267019ebe8d4402ce207b8fcc3ad578f9f96eaab7729802d3ded814 |
| PFC | libraries/PbFarming.sol | a3b326e9990f2db0101b3724b12560ccb7cd364294ca6580f1eaba7dfba8c646 |
| PPC | libraries/PbPool.sol | 6b9303c804816ebf253fbb1d1a65e7b09268e29c2c8b0c42613b62425642da1c |
| PSC | libraries/PbSgn.sol | eb157a132d45300aef958a6501f9d8891c0218dcf9f6a3e13f373a2ee05f6b1e |
| PSK | libraries/PbStaking.sol | fe04d84b66e908fc2346cf3df633f010569a0fbe5e19da2b328a06e1ecce1011 |
| FCK | test-helpers/Faucet.sol | c197173ef37726d24f544d43ef51e3913c85eac1920703faa44d485d74a67ef8 |
| MER | test-helpers/MintableERC20.sol | 1d4ceef52513a5843cfb30a9d07b29ebc5a9d905faeb4b2e6dbb24d3d86c98dd |
| TER | test-helpers/TestERC20.sol | dfe34d8aee4afb0a6428af281bbd4068ae78ae9c968f77290e65919022fe10b6 |
| BCK | Bridge.sol | 3c80d0f61157666347beff31a32e0dbd8682657fb7862b537edd17f0ec035372 |
| FRC | FarmingRewards.sol | 44f67695143b5fced7557c48c6d8fd90c9a21eacbe46734a756fd6cee3b006d2 |
| GCK | Govern.sol | 99a3813586e868b1f9669b1d6ef759831c4d16aeaf1fe668d271d1a9bbbbcf02 |
| PCP | Pool.sol | ac1ff669e9e04cdfc6cfdbf71a7ce1a7cb93d67b119769ce039b824c55f8b00d |
| SGC | SGN.sol | dfb1b697ce57be1706d87e12928d3fc69540e3c7460aea4518f978323f888eb3 |

| ID | File | SHA256 Checksum |
|---|---|---|
| SCK | Signers.sol | 9f769048bb6a734859cac0913d0c4da6703e32eb9813a51863c9076eeec2b6ee |
| SCP | Staking.sol | 1630e7d92d29db87873bcee964ece01d078e72588e636b7b6ec36cb8500a2511 |
| SRC | StakingReward.sol | d7bb33ba9a86fdd9c283de1a94e57df648797ecfcfdb7b666f635adc81f37f89 |
| VCK | Viewer.sol | f8b7fba41466a45e76d537516e098d29f50bc6b50216c9f23f3fe9051db6a6aa |
| WCK | Whitelist.sol | e38844661760e2d247e7e0179f789d1b76216c5801aa601b72e5ca1c0888f9d0 |

# Findings



| | | |
|---|---|---|
| 🟥 | **Critical** | **0** (0.00%) |
| 🟧 | **Major** | **7** (24.14%) |
| 🟨 | **Medium** | **9** (31.03%) |
| 🟨 | **Minor** | **13** (44.83%) |
| 🟦 | **Informational** | **0** (0.00%) |
| 🟩 | **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **GLOBAL-01** | Centralization Risk | **Centralization / Privilege** | ● **Major** | ⓘ Acknowledged |
| GLOBAL-02 | Validators should have the ability to stop receiving delegations and evict delegators | Logical Issue | ● Medium | ⓘ Acknowledged |
| BCK-01 | Unnecessary fallback function | Language Specific | ● Minor | ✓ Resolved |
| BCK-02 | Lack of input validation | Logical Issue | ● Minor | ⓘ Acknowledged |
| BCK-03 | Usage of `transfer()` for sending Ether | Volatile Code | ● Minor | ✓ Resolved |
| BCK-04 | Uninitialized state variable | Volatile Code | ● Medium | ⓘ Acknowledged |
| **FRC-01** | Privileged Function Allows Owner to Withdraw Tokens | **Centralization / Privilege** | ● **Major** | ⓘ Acknowledged |
| GCK-01 | Lack of input validation | Logical Issue | ● Minor | ✓ Resolved |
| PBC-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| PCK-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| PCK-02 | Lack of input validation | Logical Issue | ● Medium | ⓘ Acknowledged |
| PCK-03 | Lack of input validation | Logical Issue | ● Medium | ⓘ Acknowledged |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| PFC-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| PPC-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| PSC-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| PSK-01 | Inconsistent Solidity Version and potentially unsafe math operations | Mathematical Operations, Inconsistency | ● Minor | ✓ Resolved |
| SCK-01 | By default whitelist not enabled and hackers can block other validators from be initialized | Logical Issue | ● Major | ⓘ Acknowledged |
| SCK-02 | Malicious validator can block other validators from being initialized and updating signer | Logical Issue | ● Major | ⓘ Acknowledged |
| **SCK-03** | Privileged Function Allows Owner to Withdraw Tokens | **Centralization / Privilege** | ● **Major** | ⓘ Acknowledged |
| SCK-04 | Wrong variable used | Logical Issue | ● Medium | ✓ Resolved |
| SCK-05 | Wrong variable used | Logical Issue | ● Medium | ✓ Resolved |
| SCK-06 | Incorrect placement of the decentralization check | Logical Issue | ● Medium | ✓ Resolved |
| SCK-07 | Problematic condition check | Logical Issue | ● Minor | ✓ Resolved |
| SCK-08 | Validator status should not be `Null` | Logical Issue | ● Minor | ✓ Resolved |
| SCK-09 | `jailPeriod` should also apply when validator status is `Unbonding` | Logical Issue | ● Medium | ✓ Resolved |
| SCK-10 | Lack of input validation | Logical Issue | ● Minor | ⓘ Acknowledged |
| **SGN-01** | Privileged Function Allows Owner to Withdraw Tokens | **Centralization / Privilege** | ● **Major** | ⓘ Acknowledged |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **SRC-01** | Privileged Function Allows Owner to Withdraw Tokens | **Centralization / Privilege** | ● **Major** | ⓘ Acknowledged |
| VCK-01 | Incorrect ForLoop initial value | Logical Issue | ● Medium | ✓ Resolved |

## GLOBAL-01 | Centralization Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Major | Global | ⓘ Acknowledged |

### Description

In the contract `[Signers.sol]`, the role `[owner]` has the authority over the following function:

- [resetSigners()]
- [notifyResetSigners()]
- [increaseNoticePeriod()]

In the contract `[Bridge.sol]`, the role `[owner]` has the authority over the following function:

- [setMinSend()]
- [setMinSlippage()]
- [setWrap()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, change signers at will, make the contract invoke malicious code.

### Recommendation

We advise the client to carefully manage the `[owner]` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

### Alleviation

**[Celer team]**: "We plan to start with a guarded launch mode, for which we have a hardware wallet owner account to handle emergencies if there are any. The owner role will be revoked or transferred to a governance contract when the mainnet is proven secure and stable."

## GLOBAL-02 | Validators should have the ability to stop receiving delegations and evict delegators

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | Global | ⓘ Acknowledged |

### Description

To prevent decentralization check failure, validators should have the ability to stop receiving delegations and evict delegators. Otherwise, malicious delegators can intentionally delegate a large number of tokens such that victim validators can not become bonded.

### Recommendation

We advise the client to add functionality to `Staking.sol` to allow validators to stop receiving delegations and to evict delegators.

### Alleviation

**[Celer team]**: "This “attack” requires the malicious delegators to delegate a very large amount of tokens to an unbonded validator, just for the purpose to block a validator candidate from becoming bonded. Such an attack costs the attacker lots of liquidity, while not introducing any risk to the availability of sgn chain or fund security of any party. Therefore we believe it is less of a concern and decide to keep the code as it is."

## BCK-01 | Unnecessary fallback function

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Bridge.sol (aad51fc): 131 | ✓ Resolved |

### Description

The fallback function in the `Bridge.sol` contract is unnecessary since there is already `receive()` function to receive ethers. And fallback function will make calling non-existing functions of the contract always succeed which may not be desired behavior.

### Recommendation

Consider removing the fallback function in the `Bridge.sol` contract.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## BCK-02 | Lack of input validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Bridge.sol (aad51fc): 54 | ⓘ Acknowledged |

### Description

Based on bridge functionality, destination chain ID should not be current chain ID.

### Recommendation

We advise the client to add a check to make sure destination chain ID is not current chain ID.

### Alleviation

**[Celer team]**: "This is checked in the tendermint based SGN chan. If chain IDs are different, transfers will be refunded. We keep the code as it is to make the hardhat unitests easier."

## BCK-03 | Usage of `transfer()` for sending Ether

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Bridge.sol (aad51fc): 97 | ✓ Resolved |

### Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

### Recommendation

We advise the client to use the `sendValue()` function in Openzeppelin `Address` library. (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.3.2/contracts/utils/Address.sol)

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## BCK-04 | Uninitialized state variable

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Bridge.sol (aad51fc): 94 | ⓘ Acknowledged |

### Description

After the contract `Bridge` is deployed, by default `nativeWrap` is not initialized. It may cause malfunction of function `relay()`: sending erc20 tokens instead of chain native tokens.

### Recommendation

We advise the client to add a constructor to initialize `nativeWrap`.

### Alleviation

**[Celer team]**: "All parameters should be set correctly before the contract starts to be used (i.e., hold assets). This could be verified by anyone who wants to use the contract."

## FRC-01 | Privileged Function Allows Owner to Withdraw Tokens

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Major | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/FarmingRewards.sol (aad51fc): 94, 84, 76 | ⓘ Acknowledged |

### Description

In the contract `[FarmingRewards]`, the role `[owner]` has the authority over the following function:

- [pause()]
- [unpause()]
- [drainToken()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, steal funds from the contract.

### Recommendation

We advise the client to carefully manage the `[owner]` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

### Alleviation

**[Celer team]**: "We plan to start with a guarded launch mode, for which we have a hardware wallet owner account to handle emergencies if there are any. The owner role will be revoked or transferred to a governance contract when the mainnet is proven secure and stable."

## GCK-01 | Lack of input validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Govern.sol (aad51fc): 107 | ✓ Resolved |

### Description

The function parameter `_vote` should not be `Null`.

### Recommendation

We advise the client to add a check to make sure `_vote` is not `VoteOption.Null`.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PBC-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/PbBridge.sol (aad51fc): 5 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PCK-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/Pb.sol (aad51fc): 3 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PCK-02 | Lack of input validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/Pb.sol (aad51fc): 71~72 | ⓘ Acknowledged |

### Description

The function `decVarint()` does not check if `buf.idx` plus length of decoded data is bigger than `buf.b.length`. If it is true, the input parameter `buf` is malformed.

### Recommendation

We advise the client to add a check to make sure `buf.idx <= buf.b.length` before return.

### Alleviation

**[Celer team]**: "This does not have real risk because: 1) malformed msg won't be signed 2) overflow is checked at the beginning of each decode."

## PCK-03 | Lack of input validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/Pb.sol (aad51fc): 108~111 | ⓘ Acknowledged |

### Description

After decoding data, `buf.idx` should be equal to `end`. Otherwise, the input parameter `buf` is malformed.

### Recommendation

We advise the client to add a check to make sure `buf.idx == end` after decoding data.

### Alleviation

**[Celer team]**: "This does not have real risk because: 1) malformed msg won’t be signed 2) overflow is checked at the beginning of each decode."

## PFC-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/PbFarming.sol (aad51fc): 5 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PPC-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/PbPool.sol (aad51fc): 5 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PSC-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/PbStaking.sol (aad51fc): 5 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## PSK-01 | Inconsistent Solidity Version and potentially unsafe math operations

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations, Inconsistency | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/libraries/PbSgn.sol (aad51fc): 5 | ✓ Resolved |

### Description

The solidity versions in Pb.sol, PbBridge.sol, PbFarming.sol, PbPool.sol, PbSgn.sol, PbStaking.sol are `>=0.5.0`, while other contracts are using `0.8.9`. If the solidity version is below `0.8.0`, native math operations are not safe because overflow/underflow is not checked.

### Recommendation

It is okay to try different compiler versions during the development stage. However, we recommend locking the solidity version when it reaches the production stage, and in this case, 0.8.9 should be used.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-01 | By default whitelist not enabled and hackers can block other validators from be initialized

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Major | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 102, 110~111, 105 | ⓘ Acknowledged |

### Description

After the `Staking` contract is deployed, by default whitelist is not enabled. Anyone can call the `initializeValidator()` function and become a validator. Hackers can specify a victim validator address or the signer address of a victim validator as the `_signer` parameter. Then it will fail when the victim validator calls the function.

### Recommendation

We advise the client to enable whitelist by default, require crypto signature from signer to prove the relationship between validator and signer.

### Alleviation

**[Celer team]**: "Whitelist will be enabled right after the contract is deployed. Hackers are not incentivized to hack a contract without any tokens."

## SCK-02 | Malicious validator can block other validators from being initialized and updating signer

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Major | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 129, 133 | ⓘ Acknowledged |

### Description

A malicious validator can call function `updateValidatorSigner()` and specify a victim validator address or the signer address of a victim validator as the `_signer` parameter. Then it will fail when the victim validator tries to call function `initializeValidator()` or `updateValidatorSigner()`.

### Recommendation

We advise the client to require crypto signature from signer to prove the relationship between validator and signer.

### Alleviation

**[Celer team]**: "This could prevent some other address to become a signer, but could not pose fund security risks or system availability issues to any party."

## SCK-03 | Privileged Function Allows Owner to Withdraw Tokens

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Major | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 479, 470, 462, 454, 447, 440, 433, 425, 420 | ⓘ Acknowledged |

### Description

In the contract `[Staking.sol]`, the role `[owner]` has the authority over the following function:

- [pause()]
- [unpause()]
- [drainToken()]
- [setGovContract()]
- [setRewardContract()]
- [setWhitelistEnabled()]
- [addWhitelisted()]
- [removeWhitelisted()]
- [setMaxSlashFactor()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, steal funds from the contract.

### Recommendation

We advise the client to carefully manage the `[owner]` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

### Alleviation

**[Celer team]**: "We plan to start with a guarded launch mode, for which we have a hardware wallet owner account to handle emergencies if there are any. The owner role will be revoked or transferred to a governance contract when the mainnet is proven secure and stable."

## SCK-04 | Wrong variable used

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 579 | ✓ Resolved |

### Description

The function `getBondedValidatorsTokens()` is intended to return information about bonded validators, not all validators.

The current implementation in line 579 consist the bug and makes the function allocate excessive array memory for all validators, and many elements in the returned array are empty and wasted.

```
function getBondedValidatorsTokens() public view returns (dt.ValidatorTokens[]
memory) {
        dt.ValidatorTokens[] memory infos = new dt.ValidatorTokens[]
(valAddrs.length);
        ...
        return infos;
    }
```

### Recommendation

We advise the client to use the `bondedValAddrs.length` in line 579.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-05 | Wrong variable used

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 623 | Resolved |

### Description

In the function `getDelegatorInfo()`, the line 623 is calculating the total of `undelegationShares`, however the current implementation is using the variable `d.undelegations.queue`, which is an outdated value that might lead to incorrect result.

```
621 for (uint256 i = 0; i < len; i++) {
622     undelegations[i] = d.undelegations.queue[i + d.undelegations.head];
623     undelegationShares += d.undelegations.queue[i].shares;
624 }
625
```

### Recommendation

We advise the client to use `undelegations` instead of `d.undelegations.queue` in line 623.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-06 | Incorrect placement of the decentralization check

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 168 | ✓ Resolved |

### Description

Decentralization check should be done before the function returns because a single bonded validator should not have excessive voting power.

### Recommendation

We advise the client to call `_decentralizationCheck()` before return.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-07 | Problematic condition check

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 236 | ✓ Resolved |

### Description

If a delegator delegates 1.9 shares, then undelegates 1 share. After that he/she can never undelegate the left 0.9 share.

### Recommendation

We advise the client to modify the check to accommodate the situation describe above.

### Alleviation

Fixed in commit hash `eaae4e2d705cbbe5025c9e7b1ec9040b126122a6`.

## SCK-08 | Validator status should not be `Null`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 354 | ✓ Resolved |

### Description

Based on the context, the validator status should not be `Unbonded` or `Null`.

### Recommendation

We advise the client to add a check to make sure validator status is not `Null`.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-09 | `jailPeriod` should also apply when validator status is `Unbonding`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 363~365 | ✓ Resolved |

### Description

`jailPeriod` should also apply when validator status is `Unbonding`. Otherwise, the slashed validator can become bonded immediately.

### Recommendation

We advise the client to update `validator.bondBlock` when validator status is `Unbonding`.

### Alleviation

Fixed in commit hash `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

## SCK-10 | Lack of input validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Minor | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Staking.sol (aad51fc): 74 | ⓘ Acknowledged |

### Description

According to the comment line, the max slashing value `_maxSlashFactor` is `dt.SLASH_FACTOR_DECIMAL`, however there is no require check for ensuring the `_maxSlashFactor` range.

```
* @param _maxSlashFactor maximal slashing factor (1e6 = 100%)
```

### Recommendation

We advise the client to ensure the variable `_maxSlashFactor <= dt.SLASH_FACTOR_DECIMAL`.

### Alleviation

**[Celer team]**: "With current logic, maxSlashFactor > SLASH_FACTOR_DECIMAL is equivalent to maxSlashFactor = 100%"

## SGN-01 | Privileged Function Allows Owner to Withdraw Tokens

| Category | Severity | Location | Status |
|---|---|---|---|
| **Centralization / Privilege** | ● **Major** | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/SGN.sol (aad51fc): 101, 93, 110 | ⓘ Acknowledged |

### Description

In the contract `[SGN.sol]`, the role `[owner]` has the authority over the following function:

- [pause()]
- [unpause()]
- [drainToken()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, steal funds from the contract.

### Recommendation

We advise the client to carefully manage the `[owner]` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

### Alleviation

**[Celer team]**: "We plan to start with a guarded launch mode, for which we have a hardware wallet owner account to handle emergencies if there are any. The owner role will be revoked or transferred to a governance contract when the mainnet is proven secure and stable."

## SRC-01 | Privileged Function Allows Owner to Withdraw Tokens

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | ● Major | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/StakingReward.sol (aad51fc): 68, 60, 77 | ⓘ Acknowledged |

### Description

In the contract `[StakingReward.sol]`, the role `[owner]` has the authority over the following function:

- [pause()]
- [unpause()]
- [drainToken()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, stealing funds from the contract.

### Recommendation

We advise the client to carefully manage the `[owner]` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

### Alleviation

**[Celer team]**: "We plan to start with a guarded launch mode, for which we have a hardware wallet owner account to handle emergencies if there are any. The owner role will be revoked or transferred to a governance contract when the mainnet is proven secure and stable."

## VCK-01 | Incorrect ForLoop initial value

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Medium | projects/sgn-v2-contracts-803e5377cfe6142c049a6f5067825272a7e73b6f/contracts/Viewer.sol (aad51fc): 106 | ✓ Resolved |

### Description

If `i` starts from 1, the `tokens` value from bonded validator 0 is not checked. Then the calculation of return value may be wrong.

### Recommendation

We advise the client to use 0 as the initial value of `i`.

### Alleviation

Fixed in commit `bad4774b43c5e8abee81b9399fb3fca75d22e61e`.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# Disclaimer

AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

## About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK