Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.02.19, the SlowMist security team received the SubQuery team's security audit application for subquery-network-contracts, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

The SubQuery is a indexing tool for querying blockchains data. Anyone can build indexing service with SubQuery, and provide the api to making blockchain data easily accessible.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	The function removeRunner lacks delete logic	Others	Medium	Fixed

NO	Title	Category	Level	Status
N2	The function removeDelegation lacks cleanup logic	Others	Low	Acknowledged
N3	Suggestion that terminate could be repeated	Others	Suggestion	Acknowledged
N4	Missing event record	Malicious Event Log Audit	Suggestion	Acknowledged
N5	Risk of excessive authority	Authority Control Vulnerability Audit	Low	Acknowledged
N6	Preemptive Initialization	Race Conditions Vulnerability	Suggestion	Acknowledged
N7	Suggestions for payload messages	Others	Suggestion	Acknowledged
N8	Proposal to reduce Token quota after withdraw	Others	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/subquery/network-contracts

a6eca6b20b3e13b5bc3e167c73f31ba6683594e8

Audit scope:

contracts/ConsumerHost.sol

contracts/ConsumerRegistry.sol

contracts/IndexerRegistry.sol

contracts/ProjectRegistry.sol

contracts/PlanManager.sol

contracts/RewardsBooster.sol

contracts/RewardsDistributor.sol

contracts/RewardsHelper.sol

contracts/RewardsPool.sol

contracts/RewardsStaking.sol

contracts/ServiceAgreementRegistry.sol

contracts/Staking.sol

contracts/StakingAllocation.sol

contracts/StakingManager.sol

contracts/StateChannel.sol

contracts/l2/L2SQToken.sol

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

ConsumerHost			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setFeeRate	External	Can Modify State	onlyOwner
setControllerAccount	External	Can Modify State	-
removeControllerAccount	Public	Can Modify State	-
collectFee	External	Can Modify State	onlyOwner
getSigners	External	-	-
addSigner	External	Can Modify State	onlyOwner
removeSigner	External	Can Modify State	onlyOwner

ConsumerHost

isSigner	External	-	-
approve	External	Can Modify State	-
disapprove	External	Can Modify State	-
deposit	External	Can Modify State	-
withdraw	External	Can Modify State	-
paid	External	Can Modify State	-
claimed	External	Can Modify State	-
checkSign	External	-	-
checkSender	External	-	-
channelConsumer	External	-	-
supportsInterface	Public	-	-

ConsumerRegistry

Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
addController	External	Can Modify State	-
removeController	External	Can Modify State	-
isController	External	-	-
_isContract	Private	-	-

IndexerRegistry

Function Name	Visibility	Mutability	Modifiers

IndexerRegistry			
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setminimumStakingAmount	External	Can Modify State	onlyOwner
registerIndexer	External	Can Modify State	-
unregisterIndexer	External	Can Modify State	onlyIndexer
updateMetadata	External	Can Modify State	onlyIndexer
setControllerAccount	External	Can Modify State	onlyIndexer
isIndexer	External	-	-
getController	External	-	-
setInitialCommissionRate	Private	Can Modify State	-
setCommissionRate	External	Can Modify State	onlyIndexer
getCommissionRate	External	-	-

ProjectRegistry			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setCreatorRestricted	External	Can Modify State	onlyOwner
addCreator	External	Can Modify State	onlyOwner
removeCreator	External	Can Modify State	onlyOwner
_baseURI	Internal	-	-
_beforeTokenTransfer	Internal	Can Modify State	-
tokenURI	Public	-	-

ProjectRegistry			
supportsInterface	Public	-	-
_burn	Internal	Can Modify State	-
createProject	External	Can Modify State	-
updateProjectMetadata	External	Can Modify State	-
_updateProjectLatestDeployment	Internal	Can Modify State	-
addOrUpdateDeployment	External	Can Modify State	-
setProjectLatestDeployment	External	Can Modify State	-
startService	External	Can Modify State	onlyIndexer
stopService	External	Can Modify State	onlyIndexer
isServiceAvailable	External	-	-
getDeploymentProjectType	External	-	-
isDeploymentRegistered	Public	-	-

PlanManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setPlanLimit	External	Can Modify State	onlyOwner
createPlanTemplate	External	Can Modify State	onlyOwner
updatePlanTemplateMetadata	External	Can Modify State	onlyOwner
updatePlanTemplateStatus	External	Can Modify State	onlyOwner
createPlan	External	Can Modify State	-
removePlan	External	Can Modify State	-

PlanManager			
acceptPlan	External	Can Modify State	-
getPlan	External	-	-
getLimits	External	-	-
getPlanTemplate	Public	-	-

RewardsBooster			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setBoosterQueryRewardRate	External	Can Modify State	onlyOwner
setIssuancePerBlock	External	Can Modify State	onlyOwner
setReporter	External	Can Modify State	onlyOwner
boostDeployment	External	Can Modify State	-
removeBoosterDeployment	External	Can Modify State	-
getRunnerDeploymentBooster	Public	-	-
getAllocationRewards	External	-	-
_fixRewardsWithMissedLaborAndOverflow	Internal	-	-
_calcRewards	Private	-	-
getNewRewardsPerBooster	Public	-	-
getAccRewardsPerBooster	Public	-	-
updateAccRewardsPerBooster	Public	Can Modify State	-
getAccRewardsForDeployment	Public	-	-
onDeploymentBoosterUpdate	Public	Can Modify State	-

RewardsBooster			
onAllocationUpdate	Public	Can Modify State	-
getAccRewardsPerAllocatedToken	Public	-	-
setMissedLabor	External	Can Modify State	-
getMissedLabor	Public	-	-
_getMissedLabor	Internal	-	-
collectAllocationReward	External	Can Modify State	-
_collectAllocationReward	Internal	Can Modify State	-
getAccQueryRewardsPerBooster	Public	-	-
getQueryRewards	Public	-	-
getAccQueryRewards	Public	-	-
getBoosterQueryRewards	External	-	-
getRunnerDeploymentRewards	External	-	-
spendQueryRewards	External	Can Modify State	-
refundQueryRewards	External	Can Modify State	-

RewardsDistributor			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setLastClaimEra	External	Can Modify State	onlyRewardsStaking
setRewardDebt	External	Can Modify State	onlyRewardsStaking
resetEraReward	External	Can Modify State	onlyRewardsStaking
increaseAgreementRewards	External	Can Modify State	-

RewardsDistributor

addInstantRewards	External	Can Modify State	-
collectAndDistributeRewards	Public	Can Modify State	-
collectAndDistributeEraRewards	Public	Can Modify State	-
claim	Public	Can Modify State	-
claimFrom	Public	Can Modify State	-
_emitRewardsChangedEvent	Private	Can Modify State	-
_getCurrentEra	Private	Can Modify State	-
userRewards	Public	-	-
getRewardInfo	Public	-	-
getRewardAddTable	Public	-	-
getRewardRemoveTable	Public	-	-
getRewardDebt	Public	-	-

RewardsHelper

Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
batchApplyStakeChange	Public	Can Modify State	-
batchClaim	Public	Can Modify State	-
batchCollectAndDistributeRewards	Public	Can Modify State	-
indexerCatchup	Public	Can Modify State	-
batchCollectWithPool	Public	Can Modify State	-
getPendingStakers	Public	-	-

RewardsHelper			
getRewardsAddTable	Public	-	-
getRewardsRemoveTable	Public	-	-

RewardsPool			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setAlpha	Public	Can Modify State	onlyOwner
getReward	Public	-	-
labor	External	Can Modify State	-
collect	External	Can Modify State	-
batchCollect	External	Can Modify State	-
collectEra	External	Can Modify State	-
batchCollectEra	External	Can Modify State	-
isClaimed	External	-	-
getUnclaimDeployments	External	-	-
_batchCollect	Private	Can Modify State	-
_collect	Private	Can Modify State	-
_cobbDouglas	Private	-	-

RewardsStaking			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer

RewardsStaking			
setSettings	External	Can Modify State	onlyOwner
onStakeChange	External	Can Modify State	onlyStaking
onICRChange	External	Can Modify State	onlyIndexerRegistry
applyStakeChange	External	Can Modify State	-
applyICRChange	External	Can Modify State	-
checkAndReflectSettlement	Public	Can Modify State	-
_updateTotalStakingAmount	Private	Can Modify State	-
_getRewardsDistributor	Private	-	-
_getCurrentEra	Private	Can Modify State	-
_pendingStakeChange	Private	-	-
getTotalStakingAmount	Public	-	-
getLastSettledEra	Public	-	-
getCommissionRate	Public	-	-
getDelegationAmount	Public	-	-
getCommissionRateChangedEra	Public	-	-
getPendingStakeChangeLength	Public	-	-
getPendingStaker	Public	-	-

ServiceAgreementRegistry			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
supportsInterface	Public	-	-
setSettings	External	Can Modify State	onlyOwner

ServiceAgreementRegistry			
addEstablisher	External	Can Modify State	onlyOwner
removeEstablisher	External	Can Modify State	onlyOwner
_afterTokenTransfer	Internal	Can Modify State	-
createClosedServiceAgreement	External	Can Modify State	-
_establishServiceAgreement	Internal	Can Modify State	-
renewAgreement	External	Can Modify State	-
closedServiceAgreementExpired	Public	-	-
getClosedServiceAgreement	External	-	-
hasOngoingClosedServiceAgreement	External	-	-

Staking			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setLockPeriod	External	Can Modify State	onlyOwner
setIndexerLeverageLimit	External	Can Modify State	onlyOwner
setUnbondFeeRateBP	External	Can Modify State	onlyOwner
setMaxUnbondingRequest	External	Can Modify State	onlyOwner
reflectEraUpdate	Public	Can Modify State	-
_reflectStakingAmount	Private	Can Modify State	-
checkDelegateLimitation	External	-	onlyStakingManager
addRunner	External	Can Modify State	onlyStakingManager
removeRunner	External	Can Modify State	onlyStakingManager

Staking			
removeUnbondingAmount	External	Can Modify State	onlyStakingManager
addDelegation	External	Can Modify State	-
delegateToIndexer	External	Can Modify State	onlyStakingManager
removeDelegation	External	Can Modify State	-
_onDelegationChange	Internal	Can Modify State	-
startUnbond	External	Can Modify State	-
withdrawARequest	External	Can Modify State	onlyStakingManager
slashRunner	External	Can Modify State	onlyStakingManager
unbondCommission	External	Can Modify State	-
isEmptyDelegation	External	-	-

StakingAllocation			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
onStakeUpdate	External	Can Modify State	-
addAllocation	External	Can Modify State	-
removeAllocation	External	Can Modify State	-
runnerAllocation	External	-	-
overAllocationTime	External	-	-
isOverAllocation	External	-	-
_isAuth	Private	-	-

StakingManager			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
stake	External	Can Modify State	-
delegate	External	Can Modify State	-
unstake	External	Can Modify State	-
undelegate	External	Can Modify State	-
redelegate	External	Can Modify State	-
cancelUnbonding	External	Can Modify State	-
widthdraw	External	Can Modify State	-
slashRunner	External	Can Modify State	-
_getCurrentDelegationAmount	Internal	-	-
getTotalStakingAmount	Public	-	-
getEffectiveTotalStake	External	-	-
getAfterDelegationAmount	External	-	-
getUnbondingAmounts	External	-	-
getSlashableAmount	External	-	-

StateChannel			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setTerminateExpiration	External	Can Modify State	onlyOwner

StateChannel			
channel	External	-	-
open	External	Can Modify State	-
extend	External	Can Modify State	-
fund	External	Can Modify State	-
checkpoint	External	Can Modify State	-
terminate	External	Can Modify State	-
respond	External	Can Modify State	-
claim	External	Can Modify State	-
_checkStateSign	Private	-	-
_checkSign	Private	-	-
_settlement	Private	Can Modify State	-
_finalize	Private	Can Modify State	-
_isContract	Private	-	-

L2StandardERC20			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
supportsInterface	Public	-	-
mint	Public	Can Modify State	onlyL2Bridge
burn	Public	Can Modify State	onlyL2Bridge

4.3 Vulnerability Summary

[N1] [Medium] The function removeRunner lacks delete logic

Category: Others

Content

- contracts/Staking.sol

Deleting a runner only replaces the location, it doesn't delete the data. This means that if the next runner is not added, the failed runner will still be queried.

```
function removeRunner(address _runner) external onlyStakingManager {
    indexers[indexerNo[_runner]] = indexers[indexerLength - 1];
    indexerNo[indexers[indexerLength - 1]] = indexerNo[_runner];
    indexerLength--;
}
```

Solution

To delete useless data.

Status

Fixed; 76ee0484588cbf652836b26e1fcb5c742b7b2ce1 Fixed in this commit.

[N2] [Low] The function removeDelegation lacks cleanup logic

Category: Others

Content

- contracts/Staking.sol

The `addDelegation` logic adds `stakingIndexerNos`, `stakingIndexers`, `stakingIndexerLengths`, but the `removeDelegation` does not have a corresponding delete logic.

```
    function removeDelegation(address _source, address _runner, uint256 _amount)
external {
        require(
            msg.sender == settings.getContractAddress(SQContracts.StakingManager) ||
                msg.sender == address(this),
            'G008'
        );
        require(delegation[_source][_runner].valueAfter >= _amount && _amount > 0,
'S005');

        delegation[_source][_runner].valueAfter -= _amount;
```

```
            totalStakingAmount[_runner].valueAfter -= _amount;

            _onDelegationChange(_source, _runner);

            emit DelegationRemoved(_source, _runner, _amount);
        }
```

Solution

If the runner has exited completely then clean it up.

Status

Acknowledged; The delegation will be activated in the next era.It should be kept in order to be able to access the information properly.

[N3] [Suggestion] Suggestion that terminate could be repeated

Category: Others

Content

- contracts/StateChannel.sol

Users can repeatedly submit termination requests with the same query. Although there is no money lost in duplicate submissions, this is a bit counter-intuitive.

```
function terminate(QueryState calldata query) external {
    ChannelState storage state = channels[query.channelId];
    // check sender
    bool isIndexer = msg.sender == state.indexer;
    bool isConsumer = msg.sender == state.consumer;
    if (_isContract(state.consumer)) {
        isConsumer = IConsumer(state.consumer).checkSender(query.channelId, 
msg.sender);
    }
    require(isIndexer || isConsumer, 'G008');

    // check state
    bool allowState = state.expiredAt > block.timestamp &&
        query.spent >= state.spent &&
        query.spent < state.total;
    require(allowState, 'SC005');

    // check sign
    if (query.spent > 0) {
        bytes32 payload = keccak256(abi.encode(query.channelId, query.spent,
query.isFinal));
        _checkStateSign(query.channelId, payload, query.indexerSign,
query.consumerSign);
    } else {
        require(!query.isFinal, 'SC006');
    }

    // set state to terminate
    state.status = ChannelStatus.Terminating;
    uint256 expiration = block.timestamp + terminateExpiration;
    state.terminatedAt = expiration;
    state.terminateByIndexer = isIndexer;

    emit ChannelTerminate(query.channelId, query.spent, expiration, isIndexer);

    // update channel state.
    _settlement(query, false);
}
```

Solution

Check the status of the ChannelState. Cannot be repeated.

Status

Acknowledged

[N4] [Suggestion] Missing event record

Category: Malicious Event Log Audit

Content

The following functions in several contracts are for event logging of key parameter settings

- contracts/ConsumerHost.sol

```
setSettings
setFeeRate
addSigner
removeSigner
```

- contracts/ConsumerRegistry.sol

```
setSettings
```

- contracts/Staking.sol

```
setSettings
setLockPeriod
setIndexerLeverageLimit
setUnbondFeeRateBP
setMaxUnbondingRequest
```

- contracts/StakingManager.sol

```
setSettings
```

- contracts/StakingAllocation.sol

```
setSettings
```

- contracts/StateChannel.sol

```
setSettings
setTerminateExpiration
```

- contracts/RewardsPool.sol

```
setSettings
```

- contracts/RewardsDistributor.sol

```
setSettings
```

- contracts/ServiceAgreementRegistry.sol

```
setSettings
addEstablisher
removeEstablisher
```

- contracts/RewardsStaking.sol

```
setSettings
```

- contracts/RewardsHelper.sol

```
setSettings
```

- contracts/IndexerRegistry.sol

```
setSettings
setminimumStakingAmount
```

- contracts/ProjectRegistry.sol

```
setSettings
setCreatorRestricted
addCreator
removeCreator
```

- contracts/PlanManager.sol

```
setSettings
setPlanLimit
```

- contracts/RewardsBooster.sol

```
setSettings
setBoosterQueryRewardRate
setReporter
```

Solution

Record the corresponding event.

Status

Acknowledged

[N5] [Low] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

In this protocol the `owner` role can be reset through the `setSettings` function on the Settings contract, Settings contract is a very critical configuration parameters in this protocol protocol, if the owner of the private key leakage of the protocol will result in loss of funds, and lead to functional anomalies.

- contracts/ConsumerHost.sol

```
owner can setSettings
owner can setFeeRate
owner can addSigner
owner can removeSigner
owner can collectFee
```

- contracts/ConsumerRegistry.sol

```
owner can setSettings
```

- contracts/Staking.sol

```
owner can setSettings
owner can setLockPeriod
owner can setIndexerLeverageLimit
owner can setUnbondFeeRateBP
owner can setMaxUnbondingRequest
```

- contracts/StakingManager.sol

```
owner can setSettings
```

- contracts/StakingAllocation.sol

```
owner can setSettings
```

- contracts/StateChannel.sol

```
owner can setSettings
owner can setTerminateExpiration
```

- contracts/RewardsPool.sol

```
owner can setSettings
owner can setAlpha
```

- contracts/RewardsDistributor.sol

```
owner can setSettings
```

- contracts/RewardsStaking.sol

```
owner can setSettings
```

- contracts/RewardsHelper.sol

```
owner can setSettings
```

- contracts/IndexerRegistry.sol

```
owner can setSettings
owner can setminimumStakingAmount
```

- contracts/ProjectRegistry.sol

```
owner can setSettings
owner can setCreatorRestricted
owner can addCreator
owner can removeCreator
```

- contracts/PlanManager.sol

```
owner can setSettings
owner can setPlanLimit
owner can createPlanTemplate
owner can updatePlanTemplateMetadata
owner can updatePlanTemplateStatus
```

- contracts/RewardsBooster.sol

```
owner can setSettings
owner can setBoosterQueryRewardRate
owner can setReporter
owner can setIssuancePerBlock
```

Solution

In the short term, in order to cope with the scenario that the protocol needs to frequently set parameters in the early stage, the Admin can be divided into two roles, one is an EOA address, which is used to manage the protocol’s emergency pause permission, and the other is a multisign address, which is used to manage necessary parameter configuration and modification. This can solve the single-point risk without losing too much flexibility, but it cannot effectively mitigate the risk of excessive privileges. In the long run, it is more reasonable to entrust the protocol’s parameter configuration and modification permissions to the timelock contract, and to entrust the timelock contract to community governance can effectively mitigate the risk of excessive privileges. This can also improve the trust of community users in the protocol.

Status

Acknowledged; Has been deployed on the Base chain, using multiple signatures to minimize risk.

Settings Address:0x1d1e8C85A2C99575fCb95903C9aD9Ae2aDEA54fc

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

ConsumerHost Address:0x1185FD5a8B1dcdea654790219eAfA87105F201C5

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

ConsumerRegistry :0xd1ce436a883206a87c7e695f0d88B3b57369C477

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

Staking Address: 0x7A68b10EB116a8b71A9b6f77B32B47EB591B6Ded

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

StakingManager:0x09395a2A58DB45db0da254c7EAa5AC469D8bDc85

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

StakingAllocation Address:0x20E4B978b930ce17a499C33BbF958b5b920F70E1

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

StateChannel Address:0x6797Df373589dF2AA37FA353c4254FD7834B751A

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

RewardsPool Address:0xd2b00e427e3FE06Be815C20039421308f0487d03

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

RewardsDistributor Address:0x18AEC6c407235d446E52Aa243CD1A75421bb264e

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

RewardsStaking Address:0x1c285c5513f2135f8AD12A930E6473dA47581BE8

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

RewardsHelper Address:0x390Ef8EC1e2D90Ab7229662058B9a246bBD4Cb94

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

IndexerRegistry Address:0xadED5DDFA892250018fE54DB8E8C6CAd45476DC9

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

ProjectRegistry Address:0x5499c960cc54563E7264Fb96be4E0907a93E825B

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

PlanManager Address:0xbF443a0474AE33C30c2A0dfbc608B0e374A59DcD

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

RewardsBooster Address:0x7F138D57A5e05b6FBF3bCAdDa9a1252354245464

Owner Address: 0xDD93Add934dCc40b54f3d701C5666CFf1C9FD0Df

[N6] [Suggestion] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

The following functions may be preempted.

- contracts/ConsumerHost.sol

```
function initialize(
    ISettings _settings,
    address _sqt,
    address _channel,
    uint256 _feePerMill
) external initializer {
    __Ownable_init();
    settings = _settings;
    feePerMill = _feePerMill;

    // Approve Token to State Channel.
    IERC20 sqt = IERC20(_sqt);
    sqt.approve(_channel, sqt.totalSupply());
}
```

- contracts/ConsumerRegistry.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    settings = _settings;
}
```

- contracts/StakingManager.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    // Settings
    settings = _settings;
}
```

- contracts/Staking.sol

```
function initialize(
    ISettings _settings,
    uint256 _lockPeriod,
    uint256 _unbondFeeRate
) external initializer {
    __Ownable_init();

    indexerLeverageLimit = 10;
    maxUnbondingRequest = 20;
    unbondFeeRate = _unbondFeeRate;
    lockPeriod = _lockPeriod;
    settings = _settings;
}
```

- contracts/StakingAllocation.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    settings = _settings;
}
```

- contracts/StateChannel.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    terminateExpiration = 86400;
    settings = _settings;
}
```

- contracts/RewardsPool.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    alphaNumerator = 1;
    alphaDenominator = 3;
    settings = _settings;
}
```

- contracts/RewardsDistributor.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    //Settings
    settings = _settings;
}
```

- contracts/ServiceAgreementRegistry.sol

```
function initialize(ISettings _settings, address[] calldata _whitelist) external
initializer {
    __Ownable_init();
    __ERC721_init('SuqueryAgreement', 'SA');

    settings = _settings;
    nextServiceAgreementId = 1;

    for (uint256 i; i < _whitelist.length; i++) {
        establisherWhitelist[_whitelist[i]] = true;
    }
}
```

- contracts/RewardsStaking.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    //Settings
    settings = _settings;
}
```

- contracts/RewardsHelper.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    // Settings
    settings = _settings;
}
```

- contracts/IndexerRegistry.sol

```
function initialize(ISettings _settings, uint256 _minimumStakingAmount) external
initializer {
    __Ownable_init();

    settings = _settings;
    minimumStakingAmount = _minimumStakingAmount;
}
```

- contracts/ProjectRegistry.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();
    __ERC721_init('SubQueryProject', 'SP');
    __ERC721URIStorage_init();
    __ERC721Enumerable_init();

    settings = _settings;
    creatorRestricted[ProjectType.SUBQUERY] = true;
    creatorRestricted[ProjectType.RPC] = true;
    creatorWhitelist[msg.sender] = true;
    nextProjectId = 1;
}
```

- contracts/PlanManager.sol

```
function initialize(ISettings _settings) external initializer {
    __Ownable_init();

    settings = _settings;
    limit = 5;
    nextPlanId = 1;
}
```

- contracts/RewardsBooster.sol

```
function initialize(
    ISettings _settings,
    uint256 _issuancePerBlock,
    uint256 _minimumDeploymentBooster
) external initializer {
    __Ownable_init();

    settings = _settings;
    issuancePerBlock = _issuancePerBlock;
    minimumDeploymentBooster = _minimumDeploymentBooster;
}
```

Solution

It is suggested that the initialize operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

Status

Acknowledged

[N7] [Suggestion] Suggestions for payload messages

Category: Others

Content

- contracts/StateChannel.sol

The `payload` used in the functions `extend`, `open`, `fund`, `checkpoint`, `terminate`, and `respond` are all not ChainID, so there is a risk of replay if the contract is deployed on multiple chains.

Solution

Add ChainID to the payload.

Status

Acknowledged

[N8] [Suggestion] Proposal to reduce Token quota after withdraw

Category: Others

Content

- contracts/ConsumerHost.sol

User withdrawals do not correspond to a reduction in the amount of SQT Token given to StateChannel by the current contract.

```
function withdraw(uint256 amount) external {
    require(
        !
(IEraManager(settings.getContractAddress(SQContracts.EraManager)).maintenance()),
        'G019'
    );
    Consumer storage consumer = consumers[msg.sender];
    require(consumer.balance >= amount, 'C002');

    // transfer the balance to consumer
    IERC20(settings.getContractAddress(SQContracts.SQToken)).safeTransfer(msg.sender,
amount);
    consumer.balance -= amount;
    emit Withdraw(msg.sender, amount, consumer.balance);
}
```

Solution

User withdrawals should be followed by a corresponding reduction in the credit limit.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002402290001	SlowMist Security Team	2024.02.19 - 2024.02.29	Low Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 2 low risk, 5 suggestion vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.