Smart Contract Security Audit Report

The SlowMist Security Team received the Celer Network team's application for smart contract security audit of the SGN v2 SimpleGovernance on 2022.06.08. The following are the details and results of this smart contract security audit:

Contract Name :

SGN v2 SimpleGovernance

The contract address :

https://github.com/celer-network/sgn-v2-contracts/tree/governed-owner/contracts/governed-owner/SimpleGovernance.sol

commit: 2889dd329c592175407dd0ef89c8ef6559b95771

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

NO.	Audit Items	Result
1	Replay Vulnerability	Passed
2	Denial of Service Vulnerability	Passed
3	Race Conditions Vulnerability	Passed
4	Authority Control Vulnerability	Passed
5	Integer Overflow and Underflow Vulnerability	Passed
6	Gas Optimization Audit	Passed
7	Design Logic Audit	Passed
8	Uninitialized Storage Pointers Vulnerability	Passed
9	Arithmetic Accuracy Deviation Vulnerability	Passed

NO.	Audit Items	Result
10	"False top-up" Vulnerability	Passed
11	Malicious Event Log Audit	Passed
12	Scoping and Declarations Audit	Passed
13	Safety Design Audit	Passed
14	Non-privacy/Non-dark Coin Audit	Passed

Audit Result : Passed

Audit Number : 0X002206090003

Audit Date : 2022.06.08 - 2022.06.09

Audit Team : SlowMist Security Team

Summary conclusion : This is the SimpleGovernance contract. The voters can create and execute the proposals and add or remove voters after the votes are passed. SafeMath security module is used, which is a recommended approach. The contract does not have the Overflow and the Race Conditions issue.

The source code:

```
// SPDX-License-Identifier: GPL-3.0-only
//SlowMist// The contract does not have the Overflow and the Race Conditions issue
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

// mainly used for governed-owner to do infrequent sgn/cbridge owner operations,
// relatively prefer easy-to-use over gas-efficiency
contract SimpleGovernance {
    uint256 public constant THRESHOLD_DECIMAL = 100;
    uint256 public constant MIN_ACTIVE_PERIOD = 3600; // one hour
    uint256 public constant MAX_ACTIVE_PERIOD = 2419200; // four weeks

    using SafeERC20 for IERC20;
    enum ParamName {
        ActivePeriod,
        QuorumThreshold, // default threshold for votes to pass
        FastPassThreshold // lower threshold for less critical operations
    }

    enum ProposalType {
        ExternalDefault,
        ExternalFastPass,
        InternalParamChange,
        InternalVoterUpdate,
        InternalProxyUpdate,
        InternalTransferToken
    }

    mapping(ParamName => uint256) public params;

    struct Proposal {
        bytes32 dataHash; // hash(proposalType, targetAddress, calldata)
        uint256 deadline;
        mapping(address => bool) votes;
    }

    mapping(uint256 => Proposal) public proposals;
    uint256 public nextProposalId;

    address[] public voters;
    mapping(address => uint256) public voterPowers; // voter addr -> voting power

    // NOTE: proxies must be audited open-source non-upgradable contracts with
following requirements:
    // 1. Truthfully pass along tx sender who called the proxy function as the
governance proposer.
    // 2. Do not allow arbitrary fastpass proposal with calldata constructed by the
proxy callers.
    // See ./proxies/CommonOwnerProxy.sol for example.
    mapping(address => bool) public proposerProxies;

    event Initiated(
        address[] voters,
        uint256[] powers,
        address[] proxies,
        uint256 activePeriod,
        uint256 quorumThreshold,
        uint256 fastPassThreshold
    );

    event ProposalCreated(
        uint256 proposalId,
        ProposalType proposalType,
        address target,
        bytes data,
        uint256 deadline,
        address proposer
    );
    event ProposalVoted(uint256 proposalId, address voter, bool vote);
    event ProposalExecuted(uint256 proposalId);
    event ProposalExecutionReverted(string reason);

    event ParamChangeProposalCreated(uint256 proposalId, ParamName name, uint256
value);
    event VoterUpdateProposalCreated(uint256 proposalId, address[] voters, uint256[]
powers);
    event ProxyUpdateProposalCreated(uint256 proposalId, address[] addrs, bool[]
ops);
    event TransferTokenProposalCreated(uint256 proposalId, address receiver, address
token, uint256 amount);

    constructor(
        address[] memory _voters,
        uint256[] memory _powers,
        address[] memory _proxies,
        uint256 _activePeriod,
        uint256 _quorumThreshold,
        uint256 _fastPassThreshold
    ) {
        require(_voters.length > 0 && _voters.length == _powers.length, "invalid init
voters");
        require(_activePeriod <= MAX_ACTIVE_PERIOD && _activePeriod >=
MIN_ACTIVE_PERIOD, "invalid active period");
        require(
            _quorumThreshold < THRESHOLD_DECIMAL && _fastPassThreshold <=
_quorumThreshold,
            "invalid init thresholds"
        );
        for (uint256 i = 0; i < _voters.length; i++) {
            _addVoter(_voters[i], _powers[i]);
        }
        for (uint256 i = 0; i < _proxies.length; i++) {
            proposerProxies[_proxies[i]] = true;
        }
        params[ParamName.ActivePeriod] = _activePeriod;
        params[ParamName.QuorumThreshold] = _quorumThreshold;
        params[ParamName.FastPassThreshold] = _fastPassThreshold;
        emit Initiated(_voters, _powers, _proxies, _activePeriod, _quorumThreshold,
_fastPassThreshold);
    }

    /*********************************
     * External and Public Functions *
     *********************************/

    function createProposal(address _target, bytes memory _data) external returns
(uint256) {
        return _createProposal(msg.sender, _target, _data,
ProposalType.ExternalDefault);
    }

    // create proposal through proxy
    function createProposal(
        address _proposer,
        address _target,
        bytes memory _data,
        ProposalType _type
    ) external returns (uint256) {
        require(proposerProxies[msg.sender], "sender is not a valid proxy");
        require(_type == ProposalType.ExternalDefault || _type ==
ProposalType.ExternalFastPass, "invalid type");
        return _createProposal(_proposer, _target, _data, _type);
    }

    function createParamChangeProposal(ParamName _name, uint256 _value) external
returns (uint256) {
        bytes memory data = abi.encode(_name, _value);
        uint256 proposalId = _createProposal(msg.sender, address(0), data,
ProposalType.InternalParamChange);
        emit ParamChangeProposalCreated(proposalId, _name, _value);
        return proposalId;
    }

    function createVoterUpdateProposal(address[] calldata _voters, uint256[] calldata
_powers)
        external
        returns (uint256)
    {
        require(_voters.length == _powers.length, "voters and powers length not
match");
        bytes memory data = abi.encode(_voters, _powers);
        uint256 proposalId = _createProposal(msg.sender, address(0), data,
ProposalType.InternalVoterUpdate);
        emit VoterUpdateProposalCreated(proposalId, _voters, _powers);
        return proposalId;
    }

    function createProxyUpdateProposal(address[] calldata _addrs, bool[] calldata
_ops) external returns (uint256) {
        require(_addrs.length == _ops.length, "_addrs and _ops length not match");
        bytes memory data = abi.encode(_addrs, _ops);
        uint256 proposalId = _createProposal(msg.sender, address(0), data,
ProposalType.InternalProxyUpdate);
        emit ProxyUpdateProposalCreated(proposalId, _addrs, _ops);
        return proposalId;
    }

    function createTransferTokenProposal(
        address _receiver,
        address _token,
        uint256 _amount
    ) external returns (uint256) {
        bytes memory data = abi.encode(_receiver, _token, _amount);
        uint256 proposalId = _createProposal(msg.sender, address(0), data,
ProposalType.InternalTransferToken);
        emit TransferTokenProposalCreated(proposalId, _receiver, _token, _amount);
        return proposalId;
    }

    function voteProposal(uint256 _proposalId, bool _vote) external {
        require(voterPowers[msg.sender] > 0, "invalid voter");
        Proposal storage p = proposals[_proposalId];
        require(block.timestamp < p.deadline, "deadline passed");
        p.votes[msg.sender] = _vote;
        emit ProposalVoted(_proposalId, msg.sender, _vote);
    }

    function executeProposal(
        uint256 _proposalId,
        ProposalType _type,
        address _target,
        bytes calldata _data
    ) external {
        require(voterPowers[msg.sender] > 0, "only voter can execute a proposal");
        Proposal storage p = proposals[_proposalId];
        require(block.timestamp < p.deadline, "deadline passed");
        require(keccak256(abi.encodePacked(_type, _target, _data)) == p.dataHash,
"data hash not match");
        p.deadline = 0;

        p.votes[msg.sender] = true;
        (, , bool pass) = countVotes(_proposalId, _type);
        require(pass, "not enough votes");

        if (_type == ProposalType.ExternalDefault || _type ==
ProposalType.ExternalFastPass) {
            (bool success, bytes memory res) = _target.call(_data);
            require(success, _getRevertMsg(res));
        } else if (_type == ProposalType.InternalParamChange) {
            (ParamName name, uint256 value) = abi.decode((_data), (ParamName,
uint256));
            if (name == ParamName.ActivePeriod) {
                require(value <= MAX_ACTIVE_PERIOD && value >= MIN_ACTIVE_PERIOD,
"invalid active period");
            } else if (name == ParamName.QuorumThreshold || name ==
ParamName.FastPassThreshold) {
                require(value < THRESHOLD_DECIMAL && value > 0, "invalid threshold");
            }
            params[name] = value;
        } else if (_type == ProposalType.InternalVoterUpdate) {
            (address[] memory addrs, uint256[] memory powers) = abi.decode((_data),
(address[], uint256[]));
            for (uint256 i = 0; i < addrs.length; i++) {
                if (powers[i] > 0) {
                    _addVoter(addrs[i], powers[i]);
                } else {
                    _removeVoter(addrs[i]);
                }
            }
        } else if (_type == ProposalType.InternalProxyUpdate) {
            (address[] memory addrs, bool[] memory ops) = abi.decode((_data),
(address[], bool[]));
```

```
            for (uint256 i = 0; i < addrs.length; i++) {
                if (ops[i]) {
                    proposerProxies[addrs[i]] = true;
                } else {
                    delete proposerProxies[addrs[i]];
                }
            }
        } else if (_type == ProposalType.InternalTransferToken) {
            (address receiver, address token, uint256 amount) = abi.decode((_data), 
(address, address, uint256));
            _transfer(receiver, token, amount);
        }
        emit ProposalExecuted(_proposalId);
    }

    receive() external payable {}

    /*************************
     *  Public View Functions *
     *************************/

    function getVoters() public view returns (address[] memory, uint256[] memory) {
        address[] memory addrs = new address[](voters.length);
        uint256[] memory powers = new uint256[](voters.length);
        for (uint32 i = 0; i < voters.length; i++) {
            addrs[i] = voters[i];
            powers[i] = voterPowers[voters[i]];
        }
        return (addrs, powers);
    }

    function getVote(uint256 _proposalId, address _voter) public view returns (bool)
{
        return proposals[_proposalId].votes[_voter];
    }

    function countVotes(uint256 _proposalId, ProposalType _type)
        public
        view
        returns (
            uint256,
            uint256,
            bool
        )
    {
        uint256 yesVotes;
        uint256 totalPower;
        for (uint32 i = 0; i < voters.length; i++) {
            if (getVote(_proposalId, voters[i])) {
                yesVotes += voterPowers[voters[i]];
            }
            totalPower += voterPowers[voters[i]];
        }
        uint256 threshold;
        if (_type == ProposalType.ExternalFastPass) {
            threshold = params[ParamName.FastPassThreshold];
        } else {
            threshold = params[ParamName.QuorumThreshold];
        }
        bool pass = (yesVotes >= (totalPower * threshold) / THRESHOLD_DECIMAL);
        return (totalPower, yesVotes, pass);
    }

    /*********************************
     * Internal and Private Functions *
     *********************************/

    // create a proposal and vote yes
    function _createProposal(
        address _proposer,
        address _target,
        bytes memory _data,
        ProposalType _type
    ) private returns (uint256) {
        require(voterPowers[_proposer] > 0, "only voter can create a proposal");
        uint256 proposalId = nextProposalId;
        nextProposalId += 1;
        Proposal storage p = proposals[proposalId];
        p.dataHash = keccak256(abi.encodePacked(_type, _target, _data));
        p.deadline = block.timestamp + params[ParamName.ActivePeriod];
        p.votes[_proposer] = true;
        emit ProposalCreated(proposalId, _type, _target, _data, p.deadline,
_proposer);
        return proposalId;
    }

    function _addVoter(address _voter, uint256 _power) private {
        require(_power > 0, "zero power");
        require(voterPowers[_voter] == 0, "already a voter");
        voters.push(_voter);
        voterPowers[_voter] = _power;
    }

    function _removeVoter(address _voter) private {
        require(voterPowers[_voter] > 0, "not a voter");
        uint256 lastIndex = voters.length - 1;
        for (uint256 i = 0; i < voters.length; i++) {
            if (voters[i] == _voter) {
                if (i < lastIndex) {
                    voters[i] = voters[lastIndex];
                }
                voters.pop();
                voterPowers[_voter] = 0;
                return;
            }
        }
        revert("voter not found"); // this should never happen
    }

    function _transfer(
        address _receiver,
        address _token,
        uint256 _amount
    ) private {
        if (_token == address(0)) {
            (bool sent, ) = _receiver.call{value: _amount, gas: 50000}("");
            require(sent, "failed to send native token");
        } else {
            IERC20(_token).safeTransfer(_receiver, _amount);
        }
    }

    // https://ethereum.stackexchange.com/a/83577
    // https://github.com/Uniswap/v3-
periphery/blob/v1.0.0/contracts/base/Multicall.sol
    function _getRevertMsg(bytes memory _returnData) private pure returns (string
memory) {
        // If the _res length is less than 68, then the transaction failed silently
(without a revert message)
        if (_returnData.length < 68) return "Transaction reverted silently";
        assembly {
            // Slice the sighash.
            _returnData := add(_returnData, 0x04)
        }
        return abi.decode(_returnData, (string)); // All that remains is the revert
string
    }
}
```

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.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist