Moonwell Finance - Governance Dynamic Quorum

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 18th, 2022 - September 26th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/20/2022	Gokberk Gulgun
0.2	Document Edits	09/21/2022	Gokberk Gulgun
0.3	Draft Review	09/26/2022	Gabi Urrutia
1.0	Remediation Plan	09/30/2022	Gokberk Gulgun
1.1	Remediation Plan Review	10/03/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Moonwell Finance engaged Halborn to conduct a security audit on their Governance smart contracts beginning on September 18th, 2022 and ending on September 26th, 2022. The security assessment was scoped to the smart contracts provided to the Halborn Team.

1.2 AUDIT SUMMARY

The Team at Halborn was provided one week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were addressed by the Moonwell Finance Team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart Contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions(solgraph).
- Manual Assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Static Analysis of security for scoped contract, and imported functions.(Slither)
- Dynamic Analysis (ganache-cli, brownie, hardhat).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

1. Moonwell Finance Smart Contracts

   (a) PR 80: Moonwell Finance - Moonwell Core

- INSCOPE COMMIT ID :

d248cc9a4fc08849f0a5f5d34560f7998b182d4b

FIX COMMIT ID :

- Apollo v2 - c7da88a3fe3f0062d8a83ba808b648f1da369fec

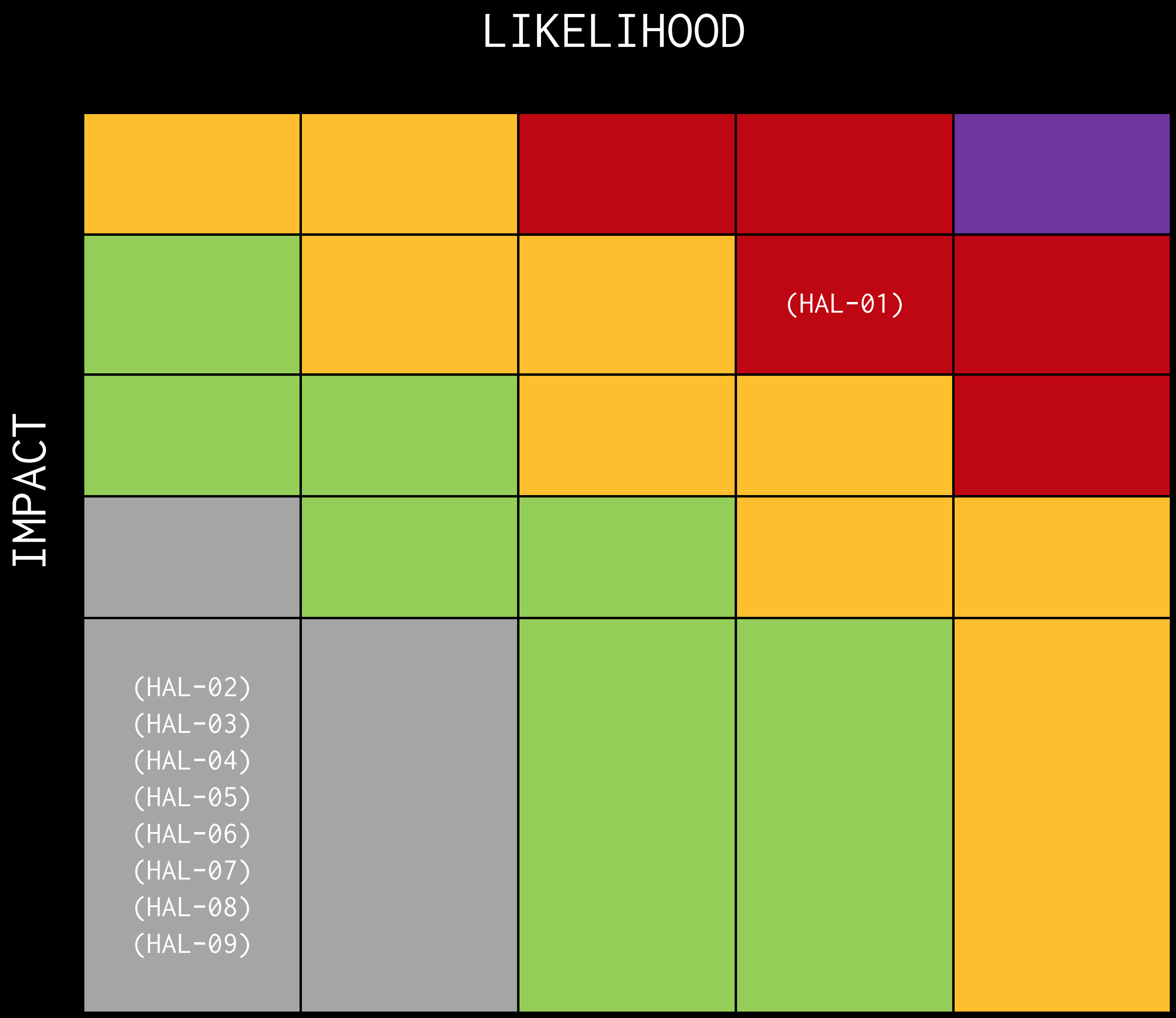
TAG :

- apollo-v2

EXECUTIVE OVERVIEW

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	0	8



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) MISSING QUORUM CAP COMPARISON CAN BREAK THE DYNAMIC QUORUM	High	SOLVED - 09/26/2022
(HAL-02) ABIENCODERV2 IS ACTIVATED BY DEFAULT 0.8+	Informational	SOLVED - 09/26/2022
(HAL-03) BUMP SOLIDITY VERSION	Informational	SOLVED - 09/26/2022
(HAL-04) NO NEED TO INITIALIZE QUORUMADJUSTED WITH FALSE	Informational	SOLVED - 09/26/2022
(HAL-05) CURRENT QUORUM CAN BE EMITTED DURING THE PROPOSAL CREATION	Informational	SOLVED - 09/26/2022
(HAL-06) USE PREFIX INCREMENT WITH THE UNCHECK CAN SAVE GAS	Informational	SOLVED - 09/26/2022
(HAL-07) SAFEMATH IS ACTIVATED BY DEFAULT AFTER 0.8.X	Informational	SOLVED - 09/26/2022
(HAL-08) MISSING NATSPEC DOCUMENTATION ON THE FUNCTIONS	Informational	SOLVED - 09/26/2022
(HAL-09) CHANGING FUNCTION VISIBILITY FROM PUBLIC TO EXTERNAL	Informational	SOLVED - 09/26/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) MISSING QUORUM CAP COMPARISON CAN BREAK THE GOVERNANCE - HIGH

Description:

GovernorApollo is implemented with a new floating quorum feature. A floating quorum is calculated as a weighted average between 80% of the old quorum and 20% of the most recent vote(s). The quorum also has upper and lower bounds. In these bounds, there is no comparison implemented. With the following scenario, dynamic quorum system can be broken.

SCENARIO:

- Set quorum caps with the timelock.
- There is no comparison between lowerQuorumCap and upperQuorumCap.
- With the timelock, lowerQuorumCap can be bigger than upperQuorumCap.
- During _calculateNewQuorum calculation, newQuorum can be directly manipulated with the timelock.

Code Location:

MoonwellApolloGovernor.sol#L523

Listing 1: MoonwellApolloGovernor.sol

```
    function setQuorumCaps(uint newLowerQuorumCap, uint newUpperQuorumCap) external {
        require(msg.sender == address(timelock), "only timelock");

        if (newLowerQuorumCap != lowerQuorumCap) {
            uint oldLowerQuorumCap = lowerQuorumCap;
            lowerQuorumCap = newLowerQuorumCap;
            emit LowerQuorumCapChanged(oldLowerQuorumCap, newLowerQuorumCap);
        }

        if (newUpperQuorumCap != upperQuorumCap) {
            uint oldUpperQuorumCap = upperQuorumCap;
            upperQuorumCap = newUpperQuorumCap;
            emit UpperQuorumCapChanged(oldUpperQuorumCap, newUpperQuorumCap);
        }
    }
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

Ensure that lowerQuorumCap is not bigger than upperQuorumCap.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by implementing the **comparison** between caps.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.2 (HAL-02) ABIENCODERV2 IS ACTIVATED BY DEFAULT 0.8+ - INFORMATIONAL

Description:

ABIEncoderV2 is being stated in a solidity version 0.8+ which is not needed since ABIEncoderV2 is activated by default 0.8+.

Code Location:

MoonwellApolloGovernor.sol#L2

Listing 2: MoonwellApolloGovernor.sol

```
pragma solidity 0.8.10;
pragma experimental ABIEncoderV2;

import "./IERC20.sol";
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the **ABIEncoderV2**.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by removing **ABIEncoderV2**.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.3 (HAL-03) BUMP SOLIDITY VERSION - INFORMATIONAL

Description:

During the review the newest version of solidity was released with the important bug fixes & Bug.

Code Location:

MoonwellApolloGovernor.sol#L2

Listing 3: MoonwellApolloGovernor.sol

```
pragma solidity 0.8.10;
pragma experimental ABIEncoderV2;

import "./IERC20.sol";
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Move from 0.8.10 to 0.8.17.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by updating pragma to **0.8.17**.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.4 (HAL-04) NO NEED TO INITIALIZE QUORUMADJUSTED WITH FALSE - INFORMATIONAL

Description:

boolean variable are initialized to a default value of false per Solidity docs. Setting a variable to the default value is unnecessary.

Code Location:

MoonwellApolloGovernor.sol#L278

Listing 4: MoonwellApolloGovernor.sol

```
        Proposal storage newProposal = proposals[proposalCount];
          newProposal.id = proposalCount;
          newProposal.proposer = msg.sender;
          newProposal.eta = 0;
          newProposal.targets = targets;
          newProposal.values = values;
          newProposal.signatures = signatures;
          newProposal.calldatas = calldatas;
          newProposal.startTimestamp = startTimestamp;
          newProposal.endTimestamp = endTimestamp;
          newProposal.startBlock = 0;
          newProposal.forVotes = 0;
          newProposal.againstVotes = 0;
          newProposal.abstainVotes = 0;
          newProposal.totalVotes = 0;
          newProposal.canceled = false;
          newProposal.executed = false;
          newProposal.quorum = currentQuorum;
          newProposal.quorumAdjusted = false;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Remove explicit initialization for default values.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by removing **explicit initialization**.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.5 (HAL-05) CURRENT QUORUM CAN BE EMITTED DURING THE PROPOSAL CREATION - INFORMATIONAL

Description:

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes with timelocks that allow users to evaluate them and consider if they would like to engage/exit based on how they perceive the changes as affecting the trustworthiness of the protocol or profitability of the implemented financial services. The alternative of directly querying on-chain contract state for such changes is not considered practical for most users/usages. In the implementation, current quorum is not emitted on the proposal generation.

Code Location:

MoonwellApolloGovernor.sol#L282

Listing 5: MoonwellApolloGovernor.sol

```
        Proposal storage newProposal = proposals[proposalCount];
        newProposal.id = proposalCount;
        newProposal.proposer = msg.sender;
        newProposal.eta = 0;
        newProposal.targets = targets;
        newProposal.values = values;
        newProposal.signatures = signatures;
        newProposal.calldatas = calldatas;
        newProposal.startTimestamp = startTimestamp;
        newProposal.endTimestamp = endTimestamp;
        newProposal.startBlock = 0;
        newProposal.forVotes = 0;
        newProposal.againstVotes = 0;
        newProposal.abstainVotes = 0;
        newProposal.totalVotes = 0;
        newProposal.canceled = false;
        newProposal.executed = false;
        newProposal.quorum = currentQuorum;
        newProposal.quorumAdjusted = false;

        latestProposalIds[newProposal.proposer] = proposalCount;

        emit ProposalCreated(newProposal.id, msg.sender, targets, values, signatures, calldatas, startTimestamp, endTimestamp, description);
        return newProposal.id;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider omitting current quorum on the proposal creation.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by adding **current quorum to the event**.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.6 (HAL-06) USE PREFIX INCREMENT WITH THE UNCHECK CAN SAVE GAS - INFORMATIONAL

Description:

The code sections use **i++** which costs more gas than **++i**, especially in a loop. Finally, the initialization of i = 0 can be skipped, as 0 is the default value.

Code Location:

MoonwellApolloGovernor.sol#L255-L676

Listing 6: MoonwellApolloGovernor.sol

```
    function getQuorum() public view returns (uint) {
        uint newQuorum = currentQuorum;

        // Start at the high water mark
        for (uint i = lastQuorumAdjustment + 1; i < proposalCount; i++) {
            // Pull state and ignore in flight proposals
            ProposalState proposalState = state(i);
            if (proposalState == ProposalState.Pending || proposalState == ProposalState.Active) {
                continue;
            }

            // Get the proposal
            Proposal storage proposal = proposals[i];

            // Only proceed if quorum for this proposal is not yet taken into account.
            if (!proposal.quorumAdjusted) {
                // If a proposal is canceled, ignore it in quorum calculations.
                if (proposalState == ProposalState.Canceled) {
                    continue;
                }

                // Adjust quorum in accordance with the proposal.
                newQuorum = _calculateNewQuorum(newQuorum, proposal.totalVotes);
            }
        }

        return newQuorum;
    }

```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use ++i instead of i++ to increment the value of an uint variable. Use unchecked where possible, and skip initialization to 0.

Remediation Plan:

SOLVED: The Moonwell team solved this issue with using prefix increment.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.7 (HAL-07) SAFEMATH IS ACTIVATED BY DEFAULT AFTER 0.8.X - INFORMATIONAL

Description:

Solidity versions >= 0.8.x perform checked arithmetic by default, so the SafeMath library is unnecessary in most cases.

Code Location:

MoonwellApolloGovernor.sol#L704-L713

Listing 7: MoonwellApolloGovernor.sol

```
    function add256(uint256 a, uint256 b) internal pure returns (uint) {
        uint c = a + b;
        require(c >= a, "addition overflow");
        return c;
    }

    function sub256(uint256 a, uint256 b) internal pure returns (uint) {
        require(b <= a, "subtraction underflow");
        return a - b;
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to delete SafeMath from the contract.

Remediation Plan:

SOLVED: The Moonwell team solved this issue with deleting **SafeMath** from the contract.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.8 (HAL-08) MISSING NATSPEC DOCUMENTATION ON THE FUNCTIONS - INFORMATIONAL

Description:

Some functions are missing @param for some of their parameters. Given that NatSpec is an important part of code documentation, this affects code comprehension, auditability, and usability.

Code Location:

MoonwellApolloGovernor.sol#L523

Listing 8: MoonwellApolloGovernor.sol

```
    function setQuorumCaps(uint newLowerQuorumCap, uint newUpperQuorumCap) external {
        require(msg.sender == address(timelock), "only timelock");

        if (newLowerQuorumCap != lowerQuorumCap) {
            uint oldLowerQuorumCap = lowerQuorumCap;
            lowerQuorumCap = newLowerQuorumCap;
            emit LowerQuorumCapChanged(oldLowerQuorumCap, newLowerQuorumCap);
        }

        if (newUpperQuorumCap != upperQuorumCap) {
            uint oldUpperQuorumCap = upperQuorumCap;
            upperQuorumCap = newUpperQuorumCap;
            emit UpperQuorumCapChanged(oldUpperQuorumCap, newUpperQuorumCap);
        }
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider adding in full NatSpec comments for all functions to have complete code documentation for future use.

Remediation Plan:

SOLVED: The Moonwell team solved this issue by adding **natspecs on the functions**.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

3.9 (HAL-09) CHANGING FUNCTION VISIBILITY FROM PUBLIC TO EXTERNAL - INFORMATIONAL

Description:

There are the functions declared as public that are never called internally within the contract. It is good practice to mark such functions as external, as this saves gas (Especially in the case where the function takes arguments, since external functions can read arguments directly from call data instead of having to allocate memory).

Code Location:

Listing 9: MoonwellApolloGovernor.sol

```
    function castVote(uint proposalId, uint8 voteValue) public {
        return _castVote(msg.sender, proposalId, voteValue);
    }

   function castVoteBySig(uint256 proposalId, uint8 voteValue, uint8 v, bytes32 r, bytes32 s) public {
        bytes32 domainSeparator = keccak256(abi.encode(DOMAIN_TYPEHASH, keccak256(bytes(name)), getChainId(), address(this)));
        bytes32 structHash = keccak256(abi.encode(BALLOT_TYPEHASH, proposalId, voteValue));
        bytes32 digest = keccak256(abi.encodePacked("\x19\x01", domainSeparator, structHash));
        address signatory = ecrecover(digest, v, r, s);
        require(signatory != address(0), "GovernorApollo::castVoteBySig: invalid signature");
        return _castVote(signatory, proposalId, voteValue);
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Functions should be marked as an external for gas optimization.

Listing 10

```
public - everyone can access.

external - Cannot be accessed internally, only externally.

internal - only this contract and contracts derived from it can access.

private - can only be accessed from this contract.
```

Remediation Plan:

SOLVED: The Moonwell team solved this issue by setting **external** functions.

Commit ID: c7da88a3fe3f0062d8a83ba808b648f1da369fec

THANK YOU FOR CHOOSING

HALBORN