// HALBORN

# Moonwell Finance - Governance & Timelock Updates

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 10th, 2022 - August 17th, 2022**

Visit: **Halborn.com**

## DOCUMENT REVISION HISTORY

| VERSION | MODIFICATION | DATE | AUTHOR |
|---|---|---|---|
| 0.1 | Document Creation | 08/18/2022 | Gokberk Gulgun |
| 0.2 | Document Edits | 08/18/2022 | Gokberk Gulgun |
| 0.3 | Draft Review | 08/18/2022 | Gabi Urrutia |
| 1.0 | Remediation Plan | 08/24/2022 | Gokberk Gulgun |
| 1.1 | Remediation Plan Review | 08/24/2022 | 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 |

# EXECUTIVE OVERVIEW

## 1.1 INTRODUCTION

Moonwell Finance engaged Halborn to conduct a security audit on their Governance & Timelock smart contracts beginning on August 10th, 2022 and ending on August 17th, 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 60: Moonwell Finance - Moonwell Core
   (b) PR 66: Moonwell Finance - Moonwell Core
   (c) PR 67: Moonwell Finance - Moonwell Core
   (d) PR 68: Moonwell Finance - Moonwell Core
   (e) PR 70: Moonwell Finance - Moonwell Core
   (f) PR 71: Moonwell Finance - Moonwell Core

**FIX COMMIT ID :**

- Artemis v2 - 4e8bec5926339106c225d0f85120ba182e52f2dd

**TAG :**

- artemis-v2

## 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| 0 | 0 | 2 | 0 | 3 |

LIKELIHOOD

IMPACT

| | | | | |
|---|---|---|---|---|
| 🟨 | 🟨 | 🟥 | 🟥 | 🟪 |
| 🟩 | 🟨 (HAL-01) | 🟨 | 🟥 | 🟥 |
| 🟩 | 🟩 | 🟨 (HAL-02) | 🟨 | 🟥 |
| ⬜ | 🟩 | 🟩 | 🟨 | 🟨 |
| ⬜ (HAL-03) (HAL-04) (HAL-05) | ⬜ | 🟩 | 🟩 | 🟨 |

| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
|---|---|---|
| (HAL-01) OVERPRIVILEGED ROLE ON THE BREAK GLASS GUARDIAN | Medium | SOLVED - 08/18/2022 |
| (HAL-02) TIMELOCK DELAY IS SET TO ZERO IN THE CONSTRUCTOR | Medium | SOLVED - 08/18/2022 |
| (HAL-03) MISSING EVENTS FOR ADMIN ONLY FUNCTIONS THAT CHANGE CRITICAL PARAMETERS | Informational | SOLVED - 08/18/2022 |
| (HAL-04) PLACE VARIABLE DEFINITION AT THE BEGINNING OF THE CONTRACT | Informational | SOLVED - 08/18/2022 |
| (HAL-05) CHANGING FUNCTION VISIBILITY FROM PUBLIC TO EXTERNAL | Informational | SOLVED - 08/18/2022 |

# FINDINGS & TECH DETAILS

## 3.1 (HAL-01) OVERPRIVILEGED ROLE ON THE BREAK GLASS GUARDIAN - MEDIUM

Description:

In the contract GovernorAlpha, the breakGlassGuardian has the authority to call the following functions to update the timelock admin:

- GovernorAlpha.__acceptAdminOnTimelock() : The breakGlassGuardian can accept address to be a timelock admin.
- GovernorAlpha.__executeSetTimelockPendingAdmin(address) : The breakGlassGuardian can add a pending admin.

Any compromise to the breakGlassGuardian account may allow the hacker to tamper with the project through these functions.

Code Location:

Listing 1: GovernorAlpha.sol

```
    function __acceptAdminOnTimelock() public {
        require(msg.sender == breakGlassGuardian, "GovernorAlpha::
↳ __acceptAdmin: sender must be bg guardian");
        timelock.acceptAdmin();
    }

    /// @notice Fast tracks setting a pendingAdmin on the timelock
↳ . Only callable by the break glass guardian.
    function __executeSetTimelockPendingAdmin(address
↳ newPendingAdmin) public {
        require(msg.sender == breakGlassGuardian, "GovernorAlpha::
↳ __executeSetTimelockPendingAdmin: sender must be bg guardian");
        timelock.fastTrackExecuteTransaction(address(timelock), 0,
↳ "setPendingAdmin(address)", abi.encode(newPendingAdmin));
    }
```

Risk Level:

**Likelihood - 2**
**Impact - 4**

Recommendation:

It is recommended to remove one of the functions and carefully manage the private key of the **breakGlassGuardian** account to avoid any potential hacking risk. In general, it is strongly recommended enhancing centralized privileges or roles in the protocol through a decentralized mechanism or smart contract-based accounts with enhanced security practices, e.g., Multi-signature wallets.

Remediation Plan:

**SOLVED:** The Moonwell Team solved this issue by removing the **__executeSetTimelockPendingAdmin** function.

Commit ID: 4e8bec5926339106c225d0f85120ba182e52f2dd

## 3.2 (HAL-02) TIMELOCK DELAY IS SET TO ZERO IN THE CONSTRUCTOR - MEDIUM

### Description:

The timelock delay is set to zero in the constructor. That can cause inconsistency in the proposals, and each proposal can bypass the timelock.

### Code Location:

Listing 2: Timelock.sol

```
    constructor(address admin_, uint delay_) public {
        require(delay_ >= MINIMUM_DELAY, "Timelock::constructor:
↳ Delay must exceed minimum delay.");
        require(delay_ <= MAXIMUM_DELAY, "Timelock::setDelay:
↳ Delay must not exceed maximum delay.");
        admin = admin_;
        delay = 0;
    }
```

### Risk Level:

**Likelihood - 3**
**Impact - 3**

### Recommendation:

It is recommended to set a delay in the constructor.

### Remediation Plan:

**SOLVED:** The Moonwell Team solved this issue by setting the **delay** function.

Commit ID: 4e8bec5926339106c225d0f85120ba182e52f2dd

## 3.3 (HAL-03) MISSING EVENTS FOR ADMIN ONLY FUNCTIONS THAT CHANGE CRITICAL PARAMETERS - INFORMATIONAL

Description:

Role-only privileged functions that change critical parameters should emit events. Events allow changing parameters to be captured so that off-chain tools/interfaces can record such changes with timelocks allowing users to evaluate them and consider whether they would like to engage/exit based on how they perceive the changes to affect reliability of the protocol or profitability of implemented financial services. The alternative of directly querying the state of the on-chain contract for such changes is not considered practical for most users/usages.

Code Location:

Listing 3: GovernorAlpha.sol

```
    function setProposalMaxOperations(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        proposalMaxOperations = newValue;
    }

    /// @notice The delay before voting on a proposal may take
 ↳ place, once proposed
    uint public votingDelay = 1 days;

    function setVotingDelay(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingDelay = newValue;
    }

    /// @notice The duration of voting on a proposal, in blocks
    uint public votingPeriod = 3 days;
    function setVotingPeriod(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingPeriod = newValue;
    }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

Add events to all admin/privileged functions that change critical parameters.

Remediation Plan:

**SOLVED:** The Moonwell Team solved this issue by adding **events** to functions.

Commit ID: 4e8bec5926339106c225d0f85120ba182e52f2dd

## 3.4 (HAL-04) PLACE VARIABLE DEFINITION AT THE BEGINNING OF THE CONTRACT - INFORMATIONAL

Description:

Regarding Solidity Style Guide, the variable definition can be moved to the beginning of the contract.

Code Location:

Listing 4: GovernorAlpha.sol

```
    function setProposalMaxOperations(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        proposalMaxOperations = newValue;
    }

    /// @notice The delay before voting on a proposal may take
↳ place, once proposed
    uint public votingDelay = 1 days;

    function setVotingDelay(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingDelay = newValue;
    }

    /// @notice The duration of voting on a proposal, in blocks
    uint public votingPeriod = 3 days;
    function setVotingPeriod(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingPeriod = newValue;
    }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

Consider moving the variable definition to the beginning of the contract.

Remediation Plan:

**SOLVED:** The Moonwell Team solved this issue by placing **variables** at the beginning of the contract.

Commit ID: 4e8bec5926339106c225d0f85120ba182e52f2dd

## 3.5 (HAL-05) 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 calldata instead of having to allocate memory).

Code Location:

Listing 5: GovernorAlpha.sol

```
    function setProposalMaxOperations(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        proposalMaxOperations = newValue;
    }

    /// @notice The delay before voting on a proposal may take
↳ place, once proposed
    uint public votingDelay = 1 days;

    function setVotingDelay(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingDelay = newValue;
    }

    /// @notice The duration of voting on a proposal, in blocks
    uint public votingPeriod = 3 days;
    function setVotingPeriod(uint newValue) public {
        require(msg.sender == address(timelock), "only timelock");
        votingPeriod = newValue;
    }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

Functions should be marked as an external for gas optimization.

**Listing 6**

```
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: 4e8bec5926339106c225d0f85120ba182e52f2dd

THANK YOU FOR CHOOSING

// HALBORN