Byte Masons - Reliquary V2

Audit Report

Version 1.1

Zigtur

July 11, 2024

Byte Masons - Reliquary V2 - Audit Report

Zigtur

July 10, 2024

Prepared by: Zigtur

Table of Contents

Introduction

Disclaimer

A smart contract security review cannot guarantee the complete absence of vulnerabilities. This effort, bound by time, resources, and expertise, aims to identify as many security issues as possible. However, there is no assurance of 100% security post-review, nor is there a guarantee that the review will uncover all potential problems in the smart contracts. It is highly recommended to conduct subsequent security reviews, implement bug bounty programs, and perform on-chain monitoring.

About Zigtur

Zigtur is an independent blockchain security researcher dedicated to enhancing the security of the blockchain ecosystem. With a history of identifying numerous security vulnerabilities across various protocols in public audit contests and private audits, **Zigtur** strives to contribute to the safety and reliability of blockchain projects through meticulous security research and reviews. Explore previous work here or reach out on X @zigtur.

About Reliquary

Reliquary is an incentive distribution system that allows users to accrue value on their position based on its age.

Users can have transferable positions that hold deposited tokens; shift deposits from one position to another; merge different positions and split a position into two.

A rehypothecation system takes users' deposits and redeposits them into a liquidity provisining platform to accrue rewards. This rehypothecation system has been adapted to support another platform: RAMSES.

Security Assessment Summary

Review commit hash - bf21cf0

Fixes review commit hash - 3256226

Deployment chains

- All EVM chains/rollups

Scope

The audit focuses mainly on the modifications introduced to support RAMSES. These modifications are described in the Github branch comparison link: aab807d…bf21cf0.

The following files are in scope of the review:

- Reliquary.sol
- libraries/DoubleStakingLogic.sol
- interfaces/IGauge.sol
- interfaces/IVoter.sol

Risk Classification

	Impact: High	**Impact:** Medium	**Impact:** Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Issues

HIGH-01 - Rewards claiming can be used to transfer any ERC-20 tokens out of `Reliquary`

Description

Scope:

- Reliquary.sol#L820-L823
- DoubleStakingLogic.sol#L102-L106

`claimGaugeRewards` function allows anyone to claim rewards allocated to the `Reliquary` contract. `Reliquary` calls `Voter.claimRewards` to claim rewards. Then, the `Reliquary` balance (which includes the claimed rewards) is transferred to the `gaugeRewardReceiver`.

However, the recent modifications now allows the caller to choose the `_rewardsTokens` that will be claimed and transferred to the `gaugeRewardReceiver`. Moreover, `Voter.claimRewards` will not revert when a reward token is not supported.

As the whole `Reliquary`'s balance is sent and the caller is able to select all the rewards tokens, the caller can transfer any ERC-20 tokens owned by the `Reliquary` contract to the `gaugeRewardReceiver`.

This issue is highly problematic if the `Reliquary` contract is supposed to hold funds such as LP tokens.

Impact

Anyone can transfer any ERC-20 tokens held by `Reliquary` to the `gaugeRewardReceiver`.

Proof of Concept

A patch to import a Foundry unit test POC is given in Appendix.

Recommendation

This issue can be fixed by implemeting an allowlist or a blocklist for reward tokens. An allowlist will be more restrictive than a blocklist.

A patch implementing a blocklist mechanism for reward tokens is available in Appendix. This blocklist will avoid the transfer of admin-configured ERC-20 tokens.

Another way to fix the issue would be to transfer only the difference in token balance.

Resolution

Byte Masons team: Fixed. Another approach than the recommendation has been applied. We just check balances before/after to check how much is owed in rewards.

Zigtur: Fix reviewed and acknowledged.

LOW-01 - Operator can transfer any ERC-20 tokens out of `Reliquary`

Description

Scope:

- Reliquary.sol#L811-L813
- DoubleStakingLogic.sol#L73-L79

The `disableGauge` function is affected by the same issue than HIGH-01.

However, `disableGauge` requires to be called by the `OPERATOR` role. This highly reduces the likelihood, downgrading the severity of this issue.

Recommendation

Implement a fix similar to the one of HIGH-01, but on the `disableGauge` function.

The patch given in Appendix for HIGH-01 also fixes this issue.

Note: this fix may not work as expected when `rewardsToken == gauge`. This is not likely to happen.

Resolution

Byte Masons team: Acknowledged.

Zigtur: Acknowledged.

INFO-01 - Incorrect comment in `updatePoolWithGaugeDeposit`

Description

Scope:

- DoubleStakingLogic.sol#L13

The comment indicates:

```
    // @dev Deposit LP tokens to earn THE.
    function updatePoolWithGaugeDeposit(
```

A similar comment was fixed in the recent modifications on the `Reliquary.sol` file (see diff here), but this was not fixed in `DoubleStakingLogic.sol`.

Recommendation

Fix the comment with the following.

```
    // @dev Deposit LP tokens to earn gauge rewards.
    function updatePoolWithGaugeDeposit(
```

Resolution

Byte Masons team: Fixed.

Zigtur: Fix reviewed and acknowledged.

Appendix

HIGH-01 - Fix patch

The following patch can be applied through `git apply` to import the recommended fix and the associated unit test.

```
diff --git a/contracts/Reliquary.sol b/contracts/Reliquary.sol
index d5c0e18..b162841 100644
--- a/contracts/Reliquary.sol
+++ b/contracts/Reliquary.sol
@@ -59,6 +59,9 @@ contract Reliquary is IReliquary, Multicall, ERC721,
↪  AccessControlEnumerable, Re
     /// @dev Info of each staked position.
     mapping(uint256 => PositionInfo) internal positionForId;

+    /// @dev Blocklist reward tokens
+    mapping(address => bool) internal blockedRewardToken;
+
     /**
      * @dev Constructs and initializes the contract.
      * @param _rewardToken The reward token contract address.
@@ -809,6 +812,7 @@ contract Reliquary is IReliquary, Multicall, ERC721,
↪  AccessControlEnumerable, Re
     }

     function disableGauge(uint256 _pid, address[] calldata _claimRewardsTokens)
     ↪  public onlyRole(OPERATOR) {
+        _checkRewards(_claimRewardsTokens);
         DoubleStakingLogic.disableGauge(voter, poolInfo, _pid,
         ↪  gaugeRewardReceiver, _claimRewardsTokens);
     }

@@ -819,6 +823,7 @@ contract Reliquary is IReliquary, Multicall, ERC721,
↪  AccessControlEnumerable, Re

     function claimGaugeRewards(uint256 _pid, address[] calldata _rewardTokens)
     ↪  public {
         if (paused) revert Reliquary__PAUSED();
+        _checkRewards(_rewardTokens);
         DoubleStakingLogic.claimGaugeRewards(voter, poolInfo,
         ↪  gaugeRewardReceiver, _pid, _rewardTokens);
     }

@@ -829,4 +834,23 @@ contract Reliquary is IReliquary, Multicall, ERC721,
↪  AccessControlEnumerable, Re
```

```
     function unpause() external onlyRole(OPERATOR) {
         paused = false;
     }
+
+    function blockRewards(address[] calldata _rewardsTokens) public
↪  onlyRole(OPERATOR) {
+        for (uint256 i = 0; i < _rewardsTokens.length; i++) {
+            blockedRewardToken[_rewardsTokens[i]] = true;
+        }
+    }
+
+    function unblockRewards(address[] calldata _rewardsTokens) public
↪  onlyRole(OPERATOR) {
+        for (uint256 i = 0; i < _rewardsTokens.length; i++) {
+            blockedRewardToken[_rewardsTokens[i]] = false;
+        }
+    }
+
+    function _checkRewards(address[] calldata _rewardsTokens) internal view {
+        for (uint256 i = 0; i < _rewardsTokens.length; i++) {
+            if (blockedRewardToken[_rewardsTokens[i]])
+                revert Reliquary__TOKEN_NOT_COMPATIBLE();
+        }
+    }
 }
diff --git a/test/foundry/Reliquary.t.sol b/test/foundry/Reliquary.t.sol
index a9a5285..f071c8d 100644
--- a/test/foundry/Reliquary.t.sol
+++ b/test/foundry/Reliquary.t.sol
@@ -396,6 +396,21 @@ contract GaugeRewardsTest is ERC721Holder, Test {
         console.log("reward: ", rewardToken.balanceOf(gaugeReceiver));
     }

+    function testGaugeRewardRevertsBlockedRewards() public {
+        uint256 amount = 1 ether;
+        uint256 relicId = reliquary.createRelicAndDeposit(address(this), 0,
↪  amount);
+        skip(1 days);
+        reliquary.update(relicId, address(this));
+        address[] memory rewardTokens = new address[](1);
+
+        rewardTokens[0] = address(oath);
+
+        reliquary.blockRewards(rewardTokens);
+
+        vm.expectRevert(IReliquary.Reliquary__TOKEN_NOT_COMPATIBLE.selector);
```

```
+         reliquary.claimGaugeRewards(0, rewardTokens);
+     }
+
      // function testDepositBonusRewarder() public {
      //     DepositBonusRewarder rewarder = new DepositBonusRewarder(
      //         1000 ether,
```

HIGH-01 - Proof of Concept

The following patch can be used to import the HIGH-01 PoC.

Then, use `forge test --mt testGaugeRewardPOCHigh01 -vvv`.

```
diff --git a/test/foundry/Reliquary.t.sol b/test/foundry/Reliquary.t.sol
index a9a5285..eb03e9f 100644
--- a/test/foundry/Reliquary.t.sol
+++ b/test/foundry/Reliquary.t.sol
@@ -396,6 +396,33 @@ contract GaugeRewardsTest is ERC721Holder, Test {
         console.log("reward: ", rewardToken.balanceOf(gaugeReceiver));
     }

+    function testGaugeRewardPOCHigh01() public {
+        uint256 amount = 1 ether;
+        uint256 relicId = reliquary.createRelicAndDeposit(address(this), 0,
↪  amount);
+        skip(1 days);
+        reliquary.update(relicId, address(this));
+        address[] memory rewardTokens = new address[](1);
+
+        // @POC: Initial situation
+        console.log("BEFORE - balanceOf reliquary:",
↪  oath.balanceOf(address(reliquary)));
+        uint256 balanceBefore = oath.balanceOf(address(gaugeReceiver));
+        console.log("BEFORE - balanceOf  receiver:", balanceBefore);
+        assert(oath.balanceOf(address(reliquary)) > 0);
+        assert(oath.balanceOf(address(gaugeReceiver)) == 0);
+
+        // @POC: Exploit `claimGaugeRewards`
+        // @POC: `oath` tokens are not supposed to be reward tokens.
+        rewardTokens[0] = address(oath);
+        reliquary.claimGaugeRewards(0, rewardTokens);
+
+        // @POC: Results
+        console.log("AFTER  - balanceOf reliquary:",
↪  oath.balanceOf(address(reliquary)));
+        uint256 balanceAfter = oath.balanceOf(address(gaugeReceiver));
+        console.log("AFTER  - balanceOf  receiver:", balanceAfter);
+        assert(oath.balanceOf(address(reliquary)) == 0);
+        assert(oath.balanceOf(address(gaugeReceiver)) > 0);
+    }
+
     // function testDepositBonusRewarder() public {
     //     DepositBonusRewarder rewarder = new DepositBonusRewarder(
```

```
    //         1000 ether,
```