SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared for: Mento
Prepared by: Sherlock
Lead Security Expert: cergyk
Dates Audited: March 15 - March 22, 2024
Prepared on: April 17, 2024

Introduction

Mento is a decentralized platform used to launch and operate multi-currency stable assets. Today Mento supports 4 decentralized stable assets on the Celo blockchain: cUSD (Celo Dollar), cEUR (Celo Euro), cREAL (Celo Real) and eXOF (CFA Franc), with more coming soon.

Scope

Repository: mento-protocol/mento-core

Branch: develop

Commit: d174c8a9810514e0ea0ddd67463854a2bfe80b32

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
2	0

Issues not fixed or acknowledged

Medium	High
0	0

Issue M-1: User Can Vote Even When They Have 0 Locked Mento (Edge Case)

Source: https://github.com/sherlock-audit/2024-02-mento-judging/issues/14

Found by

ComposableSecurity, HHK, Kose, ParthMandale, sakshamguruji, slylandro, y4y

Summary

There exists an edge case where the user will be withdrawing his entire locked MENTO amount and even then will be able to vote , this is depicted by a PoC to make things clearer.

Vulnerability Detail

The flow to receiving voting power can be understood in simple terms as follows ->

Users locks his MENTO and chooses a delegate-> received veMENTO which gives them(delegatee) voting power (there's cliff and slope at play too)

The veMENTO is not a standard ERC20 , it is depicted through "lines" , voting power declines (ie. slope period) with time and with time you can withdraw more of your MENTO.

The edge case where the user will be withdrawing his entire locked MENTO amount and even then will be able to vote is as follows ->

1.) User has locked his MENTO balance in the Locking.sol

2.) The owner of the contract "stops" the contract for some emergency reason.

3.) In this stopped state the user calls withdraw() which calls getAvailableForWithdraw() here https://github.com/sherlock-audit/2024-02-mento/blob/main/mento-core/contracts/governance/locking/Locking.sol#L97

4.) Since the contract is stopped , the `getAvailableForWithdraw` will return the entire locked amount of the user as withdrawable

```
function getAvailableForWithdraw(address account) public view returns (uint96) {
    uint96 value = accounts[account].amount;
    if (!stopped) {
      uint32 currentBlock = getBlockNumber();
      uint32 time = roundTimestamp(currentBlock);
      uint96 bias = accounts[account].locked.actualValue(time, currentBlock);
      value = value - (bias);
```

```
    }
    return value;
```

5.) The user receives his entire locked amount in L101.

6.) The owner "start()" the contract again

7.) Since the user's veMENTO power was not effected by the above flow , there still exists veMENTO a.k.a voting power to the delegate, and the user's delegate is still able to vote on proposals (even when the user has withdrew everything).

POC

Import console log first in the file , paste this test in the `GovernanceIntegration.t.sol`

```
function test_Poc_Stop() public {

   vm.prank(governanceTimelockAddress);
   mentoToken.transfer(alice, 10_000e18);

   vm.prank(governanceTimelockAddress);
   mentoToken.transfer(bob, 10_000e18);

   vm.prank(alice);
   locking.lock(alice, alice, 10_000e18, 1, 103);

   vm.prank(bob);
   locking.lock(bob, bob, 1500e18, 1, 103);

   vm.timeTravel(BLOCKS_DAY);

   uint256 newVotingDelay = BLOCKS_DAY;
   uint256 newVotingPeriod = 2 * BLOCKS_WEEK;
   uint256 newThreshold = 5000e18;
   uint256 newQuorum = 10; //10%
   uint256 newMinDelay = 3 days;
   uint32 newMinCliff = 6;
   uint32 newMinSlope = 12;

   vm.prank(alice);
   (
     uint256 proposalId,
     address[] memory targets,
     uint256[] memory values,
     bytes[] memory calldatas,
     string memory description
   ) = Proposals._proposeChangeSettings(
```

```
        mentoGovernor,
        governanceTimelock,
        locking,
        newVotingDelay,
        newVotingPeriod,
        newThreshold,
        newQuorum,
        newMinDelay,
        newMinCliff,
        newMinSlope
      );

    // ~10 mins
    vm.timeTravel(120);



    vm.startPrank(governanceTimelockAddress);
    locking.stop();
    vm.stopPrank();

    uint bal2 = mentoToken.balanceOf(alice);
    console.log(bal2);

    vm.startPrank(alice);
    locking.withdraw();
    vm.stopPrank();

    vm.startPrank(governanceTimelockAddress);
    locking.start();
    vm.stopPrank();

    uint bal = mentoToken.balanceOf(alice);
    console.log(bal);
    vm.prank(alice);


    console.log(mentoGovernor.castVote(proposalId, 1));
  }
```

You can see the Alice withdrew her entire locked amount and still was able to caste her vote.

Impact

User still able to vote even when the entire locked amount is withdrawn.

Code Snippet

https://github.com/sherlock-audit/2024-02-mento/blob/main/mento-core/contracts/governance/locking/Locking.sol#L111-L119

Tool used

Foundry

Recommendation

When the entire amount is withdrawn adjust the logic to remove the corresponding lines for the delegator.

Discussion

nevillehuang

Valid medium, users retain vote in an edge case scenario then holds when owners stops (pause) contract

sherlock-admin4

The protocol team fixed this issue in the following PRs/commits:
https://github.com/mento-protocol/mento-core/pull/410

sherlock-admin4

The Lead Senior Watson signed off on the fix.

Issue M-2: KYC credentials are invalid

Source: https://github.com/sherlock-audit/2024-02-mento-judging/issues/28

Found by

We noticed the Airgrab contract was using an incorrect KYC credential and after the contest ended, we discussed this with the KYC provider and updated the contracts accordingly.

Old credential: `level:plus;residency_not:ca,us`

New credential: `level:plus+liveness;citizenship_not:;residency_not:cd,cu,gb,ir,kp,ml,mm,ss,sy,us,ye`

This issue is not about the countries that are put as placeholder, it is about the level of the credential which lacks `liveness` and `citizenship_not:`

Issue is fixed in this PR:
https://github.com/mento-protocol/mento-core/pull/404/files

Please review this fix and include it in the final report if everything looks good now.

Discussion

sherlock-admin4

The protocol team fixed this issue in the following PRs/commits:
https://github.com/mento-protocol/mento-core/pull/404/files

sherlock-admin4

The Lead Senior Watson signed off on the fix.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.