Possum Portals Multi Asset V2 Security Review

Version 1.0

Jan 11, 2024

Conducted by:
Solthodox and MaslarovK, Independent Security Researcher

Table of Contents

1 About Solthodox

Solthodox is a smart contract developer and independent security researcher experienced in Solidity smart contract development and transitioning to security. With +1 year of experience in the development side, he has been joining security contests in the last few months. He also serves as a smart contract developer at Unlockd Finance, where he has been involved in building defi yield farming strategies to maximze the APY of it's users.

2 About MaslarovK

MaslarovK is an independent security researcher from Bulgaria with 3 years of experience in Web2 development. His curiosity and love for decentralisation and transparency made him transition to Web3. He has secured various protocols through public contests and private audits.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	Bera Market, NTLC
Repository	https://github.com/beramarket/beramarket-contracts
Commit hash	5b41e231f886b2fa450d1ebca050410ca789f151
Resolution 1	0b5320eed66b847b5ba8f533bad8edd5bc6a518f
Documentation	Not provided
Methods	Manual review & testing

Scope

src/V2MultiAsset/MintBurnToken.sol
src/V2MultiAsset/PortalNFT.sol
src/V2MultiAsset/PortalV2MultiAsset.sol
src/V2MultiAsset/VirtualLP.sol

Issues Found

Critical risk	0
High risk	1
Medium risk	0
Low risk	0
Informational	5

6 Findings

6.1 High risk

6.1.1 Some values are not properly updated in PortalV2MultiAsset::redeemNFTposition

Severity: *High risk*

Context: PortalV2MultiAsset.sol#L576

Description: When redeeming a position NFT user's values are updated based on this current stake position and the one held by the NFT. But some values are calculated using the current position values only, ignoring the ones from the NFT, resulting in a wrong accounting. The function calls `getUpdateAccount` to get the new values for the user BEFORE redeeming the NFT, which will lead to some differences:

```
        (
            ,
            ,
            ,
            uint256 stakedBalance,
            uint256 maxStakeDebt,
            uint256 portalEnergy,
        /// @audit this values are updated without including the NFT values
        ) = getUpdateAccount(msg.sender, 0, true);

        /// @dev Redeem the NFT and get the returned paramters
        (uint256 stakedBalanceNFT, uint256 portalEnergyNFT) = portalNFT.redeem(
            msg.sender,
            _tokenId
        );
        /// @audit add the values gotten from the NFT after that
        stakedBalance += stakedBalanceNFT;
        portalEnergy += portalEnergyNFT;
        /// @audit update accordingly
        _updateAccount(msg.sender, stakedBalance, maxStakeDebt, portalEnergy);
```

This happens because there are differences between the calcualtions that happen in `getUpdateAccount` and `redeem`:

```
            // getUpdateAccount
            // calculate earned based on the time elapsed from last udate
            portalEnergyEarned = (account.stakedBalance *
                (block.timestamp - account.lastUpdateTime) *
                1e18);

            // calculate increase because of increse in {maxLockDuration}
            portalEnergyIncrease = (account.stakedBalance *
                (maxLockDuration - account.lastMaxLockDuration) *
                1e18);

            // calculate net change adding both up
            portalEnergyNetChange =
                (portalEnergyEarned + portalEnergyIncrease) /
                (SECONDS_PER_YEAR * DECIMALS_ADJUSTMENT);

           //...
```

```
        // maxStakeDebt directly depends on stakedBalance, but doesnt include the
            stakedBalance from the NFT
        maxStakeDebt =
            (stakedBalance * maxLockDuration * 1e18) /
            (SECONDS_PER_YEAR * DECIMALS_ADJUSTMENT);

        // the portal energy increases by + portalEnergyEarned +
            portalEnergyIncrease
        portalEnergy = _isPositiveAmount
            ? account.portalEnergy +
                portalEnergyNetChange +
                portalEnergyAdjustment
            : account.portalEnergy +
                portalEnergyNetChange -
                portalEnergyAdjustment;
```

```
        // NFT::getAccount
       // portalEnergy doesnt increase because of maxLockDuration, only because of
           time elapsed
        account.portalEnergy +=
            (account.stakedBalance *
                (block.timestamp - account.mintTime) *
                1e18) /
            (SECONDS_PER_YEAR * DECIMALS_ADJUSTMENT);

        stakedBalance = account.stakedBalance;
        portalEnergy = account.portalEnergy;
```

This means the `portalEnergy` and `maxStakeDebt` are not properly updated and will be lesser than they should. Redemptions can lead to an unintentional asset lockup if the `maxLockDuration` changes between minting and redeeming the NFT.

Recommendation: Implement the following changes:

Add the values from the NFT first, and calculate the earnings and others after :

```
function redeemNFTposition(uint256 _tokenId) external {
        /// @dev Load user account storage pointer
        Account storage account = accounts[_user];
        /// @dev Redeem the NFT and get the returned paramters
        (uint256 stakedBalanceNFT, uint256 portalEnergyNFT) = portalNFT.redeem(
            msg.sender,
            _tokenId
        );
        /// add the values from the NFT
        account.stakedBalance += stakedBalanceNFT;
        portalEnergy += portalEnergyNFT;

        /// @dev Get the current state of the user Account
        (
            ,
            ,
            ,
            uint256 stakedBalance,
            uint256 maxStakeDebt,
            uint256 portalEnergy,
        ) = getUpdateAccount(msg.sender, 0, true);

        _updateAccount(msg.sender, stakedBalance, maxStakeDebt, portalEnergy);

        /// @dev Emit event that the Portal NFT was redeemed
        emit PortalNFTredeemed(msg.sender, msg.sender, _tokenId);
    }
```

Resolution: Resolved

6.2 Informational

6.2.1 `MetadataUri` could be immutable

Severity: *Informational*

Context: PortalNFT.sol#L36

Description: `MetadataUri` could be immutable or have a setter with `onlyOwner` modifier

Recommendation: Implement the following changes:

```
string private immutable metadataURI;
```

Resolution: Not resolved.

6.2.2 Inconsistent transfer method

Severity: *Informational*

Context: PortalV2MultiAsset.sol#L589

Description: In some cases `PortalV2MultiAsset` uses native `transfer` & `transferFrom` methods of IERC20 sometimes and `safeTransfer` & `safeTransferFrom` from SafeERC20 other times.

Recommendation: Consider using the

Resolution: Not resolved.

6.2.3 PSM token can be transferred to any address

Severity: *Informational*

Description: In some cases `PortalV2MultiAsset` uses native `transfer` & `transferFrom` methods of IERC20 sometimes and `safeTransfer` & `safeTransferFrom` from SafeERC20 other times.

Recommendation: Consider adding extra checks for `address(0)` and `address(this)` to prevent weird behaviours.

Resolution: Akwoledged.

6.2.4 Inconsistent naming convention

Severity: *Informational*

Context: VirtualLP.sol#L589

Description: The method `PSM_sendToPortalUser` don't use the solidity's camelcase standard.

Recommendation:

Consider renaming it to `sendPSMToPortalUser` instead.

Resolution: Not resolved

6.2.5 `AMOUNT_TO_CONVERT` cannot be changed

Severity: *Informational*

Context: VirtualLP.sol#L552

Description: When a user calls `convert` he must pay a fixed amount, but this fixed amount cannot be changed. This means that in the bad unlikely scenario of the value needing to be changed this will be an issue.

Recommendation: Consider adding a admin functoin to change it.

Resolution: Akwoledged