SHERLOCK

Security Review For Resolv

Private Contest Prepared For: **Resolv**
Lead Security Expert: **bughuntoor**
Date Audited: **November 21 - December 2, 2024**

Introduction

Infrastructure for tokenized ETH based delta-neutral strategy of Resolv

Scope

Repository: resolv-im/resolv-contracts

Branch: main

Audited Commit: a14fef4a6ea7699da6ff915fecb26215fa51fde5

Final Commit: ece571e91ed222768f6703180423f5712590597f

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

High	Medium
0	1

Issues Not Fixed or Acknowledged

High	Medium
0	0

Security experts who found valid issues

eeyore
aslanbek

jennifer37
bughuntoor

Oblivionis

Issue M-1: Users may lose funds when they deposit/wrap USR into WstUSR

Source: https://github.com/sherlock-audit/2024-11-resolv-judging/issues/10

Found by

Oblivionis, aslanbek, bughuntoor, eeyore, jennifer37

Summary

Malicious users can increase stUSR share's price and then users who deposit/wrap funds into wstUSR contract may lose their funds.

Root Cause

In StUSR:_convertToShares, we add `1000` share offset in the share's calculation. It means users can get 1000 share when they deposit 1 wei token. This will help prevent the inflation attack in StUSR. In WstUSR, we calculate wstUSR share's amount based on the stUSR share's price. And wstUSR share's price is 1000x stUSR share's price. This will cause that wstUSR token may face the inflation attack.

Considering below scenario:

1. Users withdraw all USR tokens from stUSR.
2. Alice deposits 1 wei USR into stUSR, 1000 stUSR shares are minted for Alice.
3. Alice donates (2000*1e18 - 1) USRs to increase stUSR share's price to 1 share --> 1e18 USR.
4. Bob wants to wrap his USR (500 * 1e18) to WstUSR. Based on current stUSR share's price, wstUSR contract should get 500 stUSR shares. When we calculate `wstUSRAmount`, we should divide `ST_USR_SHARES_OFFSET`. So the `wstUSRAmount` will be round down to 0. Bob will get 0 wstUSR token. Bob will lose all his money.

```
stUSR -->
 function _convertToShares(
     uint256 _underlyingTokenAmount,
     Math.Rounding _rounding
 ) internal view returns (uint256 shares) {
     return _underlyingTokenAmount.mulDiv(totalShares() + 1000,
↪   _totalUnderlyingTokens() + 1, _rounding);
 }
```

```
WstUSR -->
 function previewDeposit(uint256 _usrAmount) public view returns (uint256
↪  wstUSRAmount) {
     return IStUSR(stUSRAddress).previewDeposit(_usrAmount) / ST_USR_SHARES_OFFSET;
 }
```

```
    function wrap(uint256 _stUSRAmount, address _receiver) public returns (uint256
↪  wstUSRAmount) {
        _assertNonZero(_stUSRAmount);
        wstUSRAmount = convertToShares(_stUSRAmount);
        IERC20(stUSRAddress).safeTransferFrom(msg.sender, address(this),
↪  _stUSRAmount);
        _mint(_receiver, wstUSRAmount);

        emit Wrap(msg.sender, _receiver, _stUSRAmount, wstUSRAmount);

        return wstUSRAmount;
    }
    function convertToShares(uint256 _usrAmount) public view returns (uint256
↪  wstUSRAmount) {
@>        return IERC20Rebasing(stUSRAddress).convertToShares(_usrAmount) /
↪  ST_USR_SHARES_OFFSET;
    }
```

Internal pre-conditions

N/A

External pre-conditions

stUSR contract is empty. There is no shares. Or the malicious user own most shares.

Attack Path

1. Users withdraw all USR tokens from stUSR.
2. Alice deposits 1 wei USR into stUSR, 1000 stUSR shares are minted for Alice.
3. Alice donates (2000*1e18 - 1) USRs to increase stUSR share's price to 1 share --> 1e18 USR.
4. Bob wants to wrap his USR (500 * 1e18) to WstUSR. Based on current stUSR share's price, wstUSR contract should get 500 stUSR shares. When we calculate `wstUSRAmount`, we should divide `ST_USR_SHARES_OFFSET`. So the `wstUSRAmount` will be round down to 0. Bob will get 0 wstUSR token. Bob will lose all his money.

Impact

Users who deposit/wrap into wstUSR may lose their funds.

PoC

N/A

Mitigation

In WstUSR, use the same decimals for wstUSR share with stUSR share.

Discussion

fextr

Hey!

After discussing the issue with the team, we have decided not to resolve it, as an ”inflation attack” as described is not a viable solution economically and can be managed without changes to the code:

1. To manipulate the `stUSR` share price, an attacker would spend far more than any potential loss incurred by users. This means the attacker would effectively lose money, making such an exploit financially irrational and unlikely to occur in practice.
2. This issue can be effectively mitigated by ensuring an initial deposit is made into the `stUSR` contract by our team before the deployment of the `wstUSR` contract. This step establishes a stable baseline for the `stUSR` share price, preventing any inflation. Once this initial deposit is in place, user deposits into the `wstUSR` contract can safely proceed without exposing the system to inflation-based exploits.

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/resolv-im/resolv-contracts/pull/222

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.