Affine Restaking Security Review

Version 1.0

Feb 29, 2024

Conducted by:
MaslarovK and Solthodox, Independent Security Researchers

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	Affine Restaking
Repository	https://github.com/AffineLabs/contracts/blob/re-staking/src/vaults/restaking/AffineReStaking.sol
Commit hash	5d51ad056ddf50b7daddc0e560354d6359320378
Resolution	e60a97a73be5a34c70fbfdc73aa5723689533512
Documentation	https://docs.affinedefi.com
Methods	Manual review & testing

Scope

Issues Found

Critical risk	0
High risk	0
Medium risk	2
Low risk	0
Informational	1

6 Findings

6.1 Medium risk

6.1.1 The `AffineReStaking` contract has not been initialized as OZ intends you to do

Severity: *Medium risk*

Context: AffineReStaking.sol#L21C5-L31C6

Description: The `AffineReStaking` contract lacks two parent upgradeable contract initializations as recommended in the OZ documentation the `__AccessControl_init()` and `__Pausable_init()` functions should be included in the `initialize()` function. Otherwise, the upgradeable functionality could get DoSed. For instance not initializing `PausableUpgradeable` would make the storage not initialize properly.

```
// PausableUpgradeable.sol
  /**
   * @dev Initializes the contract in unpaused state.
   */
  function __Pausable_init() internal onlyInitializing {
      __Pausable_init_unchained();
  }

  function __Pausable_init_unchained() internal onlyInitializing {
      _paused = false;
  }
```

References: `https://forum.openzeppelin.com/t/defender-pausableupgradeable/7148`

```
    function initialize(address _governance, address _weth) external initializer {
        governance = _governance;
        WETH = IWETH(_weth);

        // All roles use the default admin role
        // Governance has the admin role and all roles
        _grantRole(DEFAULT_ADMIN_ROLE, governance);
        _grantRole(GUARDIAN_ROLE, governance);

        // upgradeable parent contracts storage not initialized
    }
```

Recommendation: Implement the following changes:

```
    function initialize(address _governance, address _weth) external initializer {
        governance = _governance;
        WETH = IWETH(_weth);

        // initialize them from the child contract as recommended by OZ
        __AccessControl_init();
        __Pausable_init();

        // All roles use the default admin role
        // Governance has the admin role and all roles
        _grantRole(DEFAULT_ADMIN_ROLE, governance);
        _grantRole(GUARDIAN_ROLE, governance);
```

```
    }
```

Resolution: Resolved

6.1.2 Malicious user can prevent governance from revoking a token

Severity: *Medium risk*

Context: AffineReStaking.sol#L70C5-L73C6

Description: Relying on `balanceOf` could potentially lead to a vulnerability where a malicious user could prevent governance from revoking a token with `revokeToken` by sending a very small amount to the contract.

PoC:

```
function testPreventRevokeToken() public {
    // deposit
    _giveERC20(address(ezEth), alice, init_assets);
    vm.startPrank(alice);
    ezEth.approve(address(reStaking), init_assets);
    reStaking.depositFor(address(ezEth), alice, init_assets);
    assertEq(reStaking.balance(address(ezEth), alice), init_assets);
    // withdraw
    reStaking.withdraw(address(ezEth), init_assets);
    // at this point contract balance is empty
    // and governor tries to revoke
    // malicious user sends 1 wei before
    ezEth.transfer(address(reStaking), 1);
    vm.stopPrank();
    // governor tries to revoke
    vm.startPrank();
    // transaction will revert
    vm.expectRevert(ReStakingErrors.NonZeroTokenBalance());
    reStaking.revokeToken(address(ezEth));
}
```

Recommendation: Allow to revoke a token, no matter the balance.Implement the following changes:

```
  function revokeToken(address _token) external onlyGovernance {
      if (!hasRole(APPROVED_TOKEN, _token)) revert ReStakingErrors.
          NotApprovedToken();
      _revokeRole(APPROVED_TOKEN, _token);
  }
```

Resolution: Resolved

6.2 Informational

6.2.1 Deprecated OZ-upgradeable versions used

Severity: *Information risk*

Context: AffineReStaking.sol#L5-L7

Description: The contract is importing deprecated OZ-upgradeable versions.

Recommendation: Consider using the latest OZ-upgradeable versions with improved storage patterns to mitigate any potential storage collision risks.

Resolution: Aknowledged