Audit of Value Token Contract

a report of findings by

Van Cam Pham, PhD

innovative fortuna iuvat

October 14th, 2020

Table of Contents

Document Info **1**

Contact **2**

Executive Summary **3**

Conclusion **3**

Findings **4**

Constructor input parameters 4

Code Readability 4

addMinter event 5

addMinter list adjustment 6

Disclaimer **6**

innovative fortuna iuvat 0

Document Info

Client	Value DeFi
Title	Smart Contract Audit
Auditors	Van Cam Pham, PhD
Reviewed By	Joel Farris
Approved By	Rasikh Morani

Contact

For more information on this report, contact The Arcadia Media Group Inc.

Rasikh Morani
(972) 543-3886
rasikh@arcadiamgroup.com
https://t.me/thearcadiagroup

Executive Summary

A Representative Party of the Value DeFi ("ValueDeFi") engaged The Arcadia Group ("Arcadia"), a software development, research and security company, to conduct a review of the following Value Liquidity Token smart contract on the YFV Finance repo at Commit #b7b36c48d6fa7ffad0baa5ee74384cc670e0a897.

ValueLiquidityToken.sol

Arcadia completed the review using various methods primarily consisting of dynamic and static analysis. This process included a line by line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

Conclusion

While most of the findings do not require any immediate action, it is strongly recommended in the future that code reviews and audits be done prior to launch to avoid situations where potentially risky issues could be public facing with limited recourse.

Findings

1. Constructor input parameters

- VALUE-1
- Severity: Informational
- Likelihood: Low
- Impact: Low

- Target: ValueLiquidityToken.sol
- Category: Informational
- Finding Type: Static

In the ValueLiquidityToken contract, the constructor takes the YFV Token's smart contract address as it's input, which is then used in the contract as the token's address-to-be-staked. The YFV token contract address should be carefully put in the deployment to ensure that the YFV contract address stays the same as the YFV contract which was deployed earlier.

Action Recommended: As the YFV token contract was deployed already, the ValueLiquidityToken can simply hardcode the YFV address inside the contract to avoid any issue in deploying the contract.

2. Code Readability

- VALUE-2
- Severity: Low
- Likelihood: Low
- Impact: Low

- Target: ValueLiquidityToken.sol
- Category: Informational
- Finding Type: Static
- Lines: 48, 53, 58, 63

Repeated require statement checking code should have a modifier in the contract. Specifically, `require(msg.sender == governance, "!governance");` is repeated in 4 functions.

```
    function setGovernance(address _governance) public {
        require(msg.sender == governance, "!governance");
        governance = _governance;
    }

    function addMinter(address _minter) public {
        require(msg.sender == governance, "!governance");
        minters[_minter] = true;
```

```
    }

    function removeMinter(address _minter) public {
        require(msg.sender == governance, "!governance");
        minters[_minter] = false;
    }

    function setCap(uint256 _cap) public {
        require(msg.sender == governance, "!governance");
        require(_cap.add(yfvLockedBalance) >= totalSupply(), "_cap (plus
yfvLockedBalance) is below current supply");
        cap = _cap;
    }
```

Action Recommended: The code should have a modifier isGovernance to check whether function callers in the functions are the governance contract. This improves readability.

3. addMinter event

- VALUE-3
- Severity: Notice
- Likelihood: N/A
- Impact: low

- Target: ValueLiquidityToken.sol
- Category: Informational
- Finding Type: Static
- Lines: 52

In the ValueLiquidityToken contract, the function addMinter does not emit any event when the function is executed. Having events emitted in the function should allow anyone to check which addresses are in the minters list.

```
  function addMinter(address _minter) public {
        require(msg.sender == governance, "!governance");
        minters[_minter] = true;
    }
```

Action Recommended: Add an event to the contract and emit it in the addMinter function.

4. addMinter list adjustment

- VALUE-4
- Severity: Medium
- Likelihood: Low
- Impact: Low
- Target: ValueLiquidityToken.sol
- Category: Security
- Finding Type: Static
- Lines: 52

In the ValueLiquidityToken contract, the function addMinter can be called by the governance, which is initially the contract deployment wallet. This function can add any address to the minters list before transferring the contract to a community-driven governance contract.

```
function addMinter(address _minter) public {
        require(msg.sender == governance, "!governance");
        minters[_minter] = true;
    }
```

Action Recommended: The governance should be transferred to a multisignature contract/community-driven contract as soon as possible after contract deployment.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or for damages resulting from the use of the provided information. Additionally Arcadia would like to emphasize that use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period of time.