HAECHI AUDIT

DFX Finance

Smart Contract Security Analysis

Published on : May 04, 2022

Version v2.0

HAECHI AUDIT

Smart Contract Audit Certificate

DFX Finance

Security Report Published by HAECHI AUDIT
v2.0 Apr 28, 2022

Auditor : Andy Koo

Andy Koo

Executive Summary

Severity of Issues	Findings	Resolved	Unresolved	Acknowledged	Comment
Critical	-	-	-	-	-
Major	3	2	-	1	-
Minor	1	1	-	-	-
Tips	2	2	-	-	-

TABLE OF CONTENTS

6 Issues (0 Critical, 3 Major, 1 Minor, 2 tips) Found.

5 Issues resolved, 1 issue acknowledged.

ABOUT US

HAECHI AUDIT believes in the power of cryptocurrency and the next paradigm it will bring.

We have the vision to *empower the next generation of finance*. By providing security and trust in the blockchain industry, we dream of a world where everyone has easy access to blockchain technology.

HAECHI AUDIT is a flagship service of HAECHI LABS, the leader of the global blockchain industry. HAECHI AUDIT provides specialized and professional smart contract security auditing and development services.

We are a team of experts with years of experience in the blockchain field and have been trusted by 300+ project groups. Our notable partners include Universe,1inch, Klaytn, Badger, etc.

HAECHI AUDIT is the only blockchain technology company selected for the Samsung Electronics Startup Incubation Program in recognition of our expertise. We have also received technology grants from the Ethereum Foundation and Ethereum Community Fund.

Inquiries: audit@haechi.io

Website: audit.haechi.io

INTRODUCTION

This report was prepared to audit the security of the ASC(dfxCAD) smart contract created by the DFX Finance team. HAECHI AUDIT conducted the audit focusing on whether the smart contract created by DFX Finance team is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the smart contract.

CRITICAL Critical issues must be resolved as critical flaws that can harm a wide range of users.

MAJOR Major issues require correction because they either have security problems or are implemented not as intended.

MINOR Minor issues can potentially cause problems and therefore require correction.

TIPS Tips issues can improve the code usability or efficiency when corrected.

HAECHI AUDIT recommends that the DFX Finance team improve all issues discovered.

The following issue explanation uses the format of {file name}#{line number}, {contract name}#{function/variable name} to specify the code. For instance, *Sample.sol:20* points to the 20th line of Sample.sol file, and *Sample#fallback()* means the fallback() function of the Sample contract.

Please refer to the Appendix to check all results of the tests conducted for this report.

SUMMARY

The codes used in this Audit can be found on the following repository and commit hash.

Repository : https://github.com/dfx-finance/asc/

Commit hash : 5ea1aa9573cba00d1b6a426c21803e3efed790c7

Issues

HAECHI AUDIT found 0 critical issues, 3 major issues, and 1 minor issues. There are 2 Tips issues explained that would improve the code's usability or efficiency upon modification.

We confirmed that 5 issues are resolved and 1 issue has been acknowledged by the DFX Finance team.

Severity	Issue	Status
⚠ **MAJOR**	Unlimited rights for the CR_DEFENDER role of the contract	(Acknowledged - v2.0)
⚠ **MAJOR**	Contract owner can withdraw reward and staking tokens	(Resolved - v2.0)
⚠ **MAJOR**	setPause implementation does not exist.	(Resolved - v2.0)
◙ **MINOR**	Mint and Burn are unavailable after deployment for an oracle update period due to an uninitialized price.	(Resolved - v2.0)
💡 **TIPS**	There are missing Events.	(Resolved - v2.0)
💡 **TIPS**	Use an unchecked block on desired overflow implementation.	(Resolved - v2.0)

OVERVIEW

Contracts subject to audit

- ASCUpgradableProxy.sol
- dfx-cadc
  - DfxCadLogicV1.sol
  - DfxCadcLogic.sol
  - DfxCadcState.sol
  - ERC20Upgradeable.sol
- interfaces
  - IChainLinkOracle.sol
  - IDfxCurve.sol
  - IDfxOracle.sol
  - IUniswapV2.sol
  - IUniswapV3.sol
- libraries
  - AddressStringUtil.sol
  - Babylonian.sol
  - BitMath.sol
  - FixedPoint.sol
  - FixedPoint96.sol
  - FullMath.sol
  - SafeCast.sol
  - SafeERC20Namer.sol
  - SqrtPrice.sol
  - TickMath.sol
  - TransferHelper.sol
  - UniswapV2.sol
  - UniswapV3.sol
  - UnsafeMath.sol
- liquidity-mining
  - StakingRewards.sol
- oracles
  - DfxCadTWAP.sol
  - UniswapV2Oracle.sol
  - UniswapV3Oracle.sol

The smart contract has the following privileges.

- ❖ SUDO_ROLE
- ❖ POKE_ROLE
- ❖ MARKET_MAKER_ROLE
- ❖ CR_DEFENDER
- ❖ OWNER

Details of the control of each privilege are as follows:

Role	Functions
SUDO_ROLE	❖ *DfxCadcLogic#setDfxCadTwap()* ❖ *DfxCadcLogic#setPokeDelta()* ❖ *DfxCadcLogic#setDfxCadTwap()* ❖ *DfxCadcLogic#setFeeRecipient()* ❖ *DfxCadcLogic#recoverERC20()* ❖ *DfxCadcLogic#setMintBurnFee()* ❖ *DfxCadcLogic#setPaused()* ❖ *DfxCadTWAP#update()* ❖ *DfxCadTWAP#setPeriod()*
POKE_ROLE	❖ *DfxCadcLogic#pokeUp()* ❖ *DfxCadcLogic#pokeDown()*
MARKET_MAKER_ROLE	❖ *Market makers don't need to pay a mint/burn fee* ❖ *DfxCadcLogic#mint()* ❖ *DfxCadcLogic#burn()*
CR_DEFENDER	❖ *Collateral defenders to perform buyback and recollateralization* ❖ *DfxCadcLogic#execute()*
OWNER	❖ *StakingRewards#notifyRewardAmount()* ❖ *StakingRewards#recoverERC20()* ❖ *StakingRewards#setRewardsDuration()*

FINDINGS

⚠ MAJOR

Unlimited rights for the CR_DEFENDER role of the contract

(Acknowledged - v.2.0)

```
function execute(address _target, bytes memory _data)
    public
    onlyRole(CR_DEFENDER)
    returns (bytes memory response)
{
    require(_target != address(0), "target-address-required");

    // call contract in current context
    assembly {
        let succeeded := delegatecall(
            sub(gas(), 5000),
            _target,
            add(_data, 0x20),
            mload(_data),
            0,
            0
        )
        let size := returndatasize()

        response := mload(0x40)
        mstore(
            0x40,
            add(response, and(add(add(size, 0x20), 0x1f), not(0x1f)))
        )
        mstore(response, size)
        returndatacopy(add(response, 0x20), 0, size)

        switch iszero(succeeded)
        case 1 {
            // throw if delegatecall failed
            revert(add(response, 0x20), size)
        }
    }
}
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/dfx-cadc/DfxCadcLogic.sol#L161-L194]

Issue

The *CR_DEFENDER* role has the capability to call the *DfxCadcLogic#execute* function, which performs an arbitrary delegatecall that can modify the contract’s current states. This function demeans the access control of other state transition functions as the delegatecall can bypass other access controls.

As the purpose of the *DfxCadcLogic#execute* function is buyback and re-collateralization, the function has excessive permission, which may result in an unintended state change.

Recommendation

Instead of using the delegatecall, implementing each function of the buyback and re-collateralization is desirable.
In case of maintaining the current *DfxCadcLogic#execute* function,

1. Strictly manage the account which has *CR_DEFENDER* using a multi-signature wallet.
2. Document clearly the known risk and the purpose of using this function.

Update

"As recommended, the CR_DEFENDER role is currently a multisig wallet controlled by the core team. The risks of this function are well understood and are only to be used in emergency situations. There are also plans to further decentralize and increase the robustness of this multisig."

⚠ MAJOR

Contract owner can withdraw reward and staking tokens

(Found - v.1.0)

```
function recoverERC20(address tokenAddress, uint256 tokenAmount)
    external
    onlyOwner
{
    IERC20(tokenAddress).safeTransfer(msg.sender, tokenAmount);
    emit Recovered(tokenAddress, tokenAmount);
}
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/liquidity-mining/StakingRewards.sol#L152-L158]

Issue

The *StakingRewards#recoverERC20* function has no token address check that protects the stakers that their staking tokens and their rewards tokens are not accessible by the owner.

Recommendation

the *StakingRewards#recoverERC20* should be reverted when the provided token address is staking/reward token.

Update

Fixed [PR Link]

⚠ MAJOR

setPause implementation does not exist.

(Found - v.1.0)

```
import "@openzeppelin/contracts/security/Pausable.sol";

contract StakingRewards is Ownable, ReentrancyGuard, Pausable {
…
}
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/liquidity-mining/StakingRewards.sol#L10]

```
function _pause() internal virtual whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}

function _unpause() internal virtual whenPaused {
    _paused = false;
    emit Unpaused(_msgSender());
}
```

[https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c239e1af8d1a1296577108dd6989a17b57434f8e/contracts/security/Pausable.sol#L75-L90]

Issue

The *StakingRewards* contract inherits Open Zeppelin's pausable contract. OZ's Pausable contract doesn't provide public/external setPause implementation, the owner can't control the pause state of the *StakingRewards#stake* function.

Recommendation

Implementing the setPause function is recommended.

Update

Fixed [PR Link]

MINOR

Mint and Burn are unavailable after deployment for an oracle update period due to an uninitialized price.

(Found - v.1.0)

```
FixedPoint.uq112x112 public price0Average;
FixedPoint.uq112x112 public price1Average;

IUniswapV2Pair public immutable pair;
address public immutable token0;
address public immutable token1;

constructor(
    address factory,
    address tokenA,
    address tokenB,
    uint256 _period
) {
    period = _period;

    pair = IUniswapV2Pair(
        IUniswapV2Factory(factory).getPair(tokenA, tokenB)
    );

    token0 = pair.token0();
    token1 = pair.token1();

    price0CumulativeLast = pair.price0CumulativeLast();
    price1CumulativeLast = pair.price1CumulativeLast();

    uint112 reserve0;
    uint112 reserve1;
    (reserve0, reserve1, blockTimestampLast) = pair.getReserves();
    require(reserve0 > 0 && reserve1 > 0, "empty-pair");
}
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/oracles/UniswapV2Oracle.sol#L20-L50]

Issue

The mint and burn functions calculate DFX token amounts using the ratio of the price between CADC and DFX. The DFX price is checked using UniswapV2 TWAP. When the *UniswapV2Oracle* is initialized, the *UniswapV2Oracle#price0Average* value is not set. In turn, the price of DFX is calculated as zero. The mint and burn function reverts because the *FullMath#mulDiv* function prevents zero denominators.

Recommendation

Initializing *price0Average* on *UniswapV2Oracle#constructor* can resolve this issue.

Update

Fixed [PR Link]

💡 **TIPS**

There are missing Events.

(Found - v.1.0)

Issue

Without Event, it is difficult to identify in real-time whether correct values are recorded on the blockchain. In this case, it becomes problematic to determine whether the corresponding value has been changed in the application and whether the corresponding function has been called.

Recommendation

We recommend adding Events corresponding to the change occurring in the function. The following state change function is recommended to emit events.

- *DfxCadcLogic#setPokeDelta*
- *DfxCadcLogic#pokeUp*
- *DfxCadcLogic#pokeDown*
- *DfxCadcLogic#setDfxCadTwap*
- *DfxCadcLogic#setFeeRecipient*
- *DfxCadcLogic#recoverERC20*
- *DfxCadcLogic#setMintBurnFee*
- *DfxCadcLogic#execute*
- *DfxCadTWAP#update*
- *DfxCadTWAP#setPeriod*

Update

Fixed [PR Link]

💡 **TIPS**

Use an unchecked block on desired overflow implementation.

(Found - v.1.0)

```
uint32 timeElapsed = blockTimestamp - blockTimestampLast;
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/oracles/UniswapV2Oracle.sol#L57]

```
uint32 timeElapsed = blockTimestamp - blockTimestampLast;
```

[https://github.com/dfx-finance/asc/blob/5ea1aa9573cba00d1b6a426c21803e3efed790c7/src/libraries/UniswapV2.sol#L29]

Issue

The Uniswap V2 oracle allows overflow on price accumulator functionality. Revert on overflow could cause a liveness failure.

Recommendation

Using an unchecked block is recommended.

Update

Fixed [PR Link]

DISCLAIMER

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects on DFX Finance. In order to write secure smart contracts, correction of discovered problems and sufficient testing thereof are required.

Appendix A. Test Results

The following results show the unit test results covering the key logic of the smart contract subject to the security audit. Parts marked in red are test cases that failed to pass the test due to existing issues.

```
# Logic and Oracle
[PASS] test_approve_allowance_check() (gas: 376129)
[PASS] test_approve_emit_event() (gas: 376266)
[PASS] test_burn_fail_amount_more_than_balance() (gas: 351978)
[PASS] test_burn_fail_paused() (gas: 399851)
[PASS] test_burn_underlying_value_check() (gas: 679221)
[PASS] test_execute_CR_DEFENDER_return_data_check() (gas: 27891)
[PASS] test_execute_fail_normal_user_call() (gas: 84523)
[PASS] test_getUnderlyings_after_first_update() (gas: 154734)
[FAIL. Reason: Revert] test_getUnderlyings_before_first_update() (gas: 68093)
[PASS] test_grantRole_fail_normal_user() (gas: 86587)
[PASS] test_grantRole_user_added_check() (gas: 57031)
[PASS] test_logic_Initialization() (gas: 93229)
[PASS] test_logic_fail_reInitialization() (gas: 28084)
[PASS] test_mint_expected_amount_should_be_minted() (gas: 342376)
[PASS] test_mint_fail_paused() (gas: 208633)
[PASS] test_pokeDown() (gas: 64681)
[PASS] test_pokeDown_fail_too_many() (gas: 94935)
[PASS] test_pokeUp() (gas: 64636)
[PASS] test_pokeUp_fail_too_many() (gas: 825700)
[PASS] test_renounceRole_normal_user_has_no_effect() (gas: 28464)
[PASS] test_revokeRole_fail_normal_user() (gas: 86620)
[PASS] test_setDfxCadTwap_fail_non_suoder() (gas: 81628)
[PASS] test_setFeeRecipient_fail_non_suoder() (gas: 81606)
[PASS] test_setMintBurnFee_fail_non_suoder() (gas: 79511)
[PASS] test_setPaused_fail_non_suoder() (gas: 79439)
[PASS] test_setPokeDelta_fail_non_suoder() (gas: 79489)
[PASS] test_transferFrom_balance_check() (gas: 409576)
[PASS] test_transferFrom_fail_amount_more_than_allowance() (gas: 374826)
[PASS] test_transferFrom_fail_amount_more_than_balance() (gas: 377462)
[PASS] test_transferFrom_fail_recipient_zero_address() (gas: 375115)
[PASS] test_transferFrom_same_sender_and_recipient_has_no_effects() (gas: 387433)
[PASS] test_transfer_balance_check() (gas: 380626)
[PASS] test_transfer_emit_event() (gas: 374753)
[PASS] test_transfer_fail_more_than_balance() (gas: 350607)
[PASS] test_transfer_fail_zero_address() (gas: 346121)
[PASS] test_twap_grantRole_fail_normal_user() (gas: 78468)
[PASS] test_twap_grantRole_user_added_check() (gas: 47308)
[PASS] test_twap_renounceRole_normal_user_has_no_effect() (gas: 18570)
```

```
[PASS] test_twap_revokeRole_fail_normal_user() (gas: 78377)
[PASS] test_twap_setPeriod_fail_non_sudoer() (gas: 71929)
[PASS] test_twap_update_before_period() (gas: 99174)
[PASS] test_twap_update_fail_non_sudoer() (gas: 71933)


# Staking Rewards
[PASS] test_constructor_check() (gas: 20270)
[PASS] test_earned_increase_after_stake_check() (gas: 204279)
[PASS] test_exit_retrieve_earned_and_increase_rewards_balance() (gas: 315382)
[PASS] test_getRewardForDuration_increase_rewardToken_balance_check() (gas: 119964)
[PASS] test_getReward_increase_rewardToken_balance_check() (gas: 290271)
[PASS] test_lastTimeRewardApplicable_should_be_equal_current_timestamp_when_updated() (gas:
103674)
[PASS] test_notifyRewardAmount_fail_non_admin() (gas: 15401)
[PASS] test_notifyRewardAmount_fail_reward_greater_than_balance() (gas: 114408)
[PASS] test_recoverERC20_balance_check() (gas: 34894)
[PASS] test_recoverERC20_emit_event() (gas: 32642)
[PASS] test_recoverERC20_fail_non_admin() (gas: 16543)
[FAIL. Reason: Call did not revert as expected] test_recoverERC20_fail_rewardToken() (gas: 60788)
[FAIL. Reason: Call did not revert as expected] test_recoverERC20_fail_stakingToken() (gas: 46755)
[PASS] test_rewardPerToken_update_check() (gas: 205318)
[PASS] test_rewardRate_increase_when_new_rewards_added_before_duration() (gas: 230724)
[PASS] test_rewardRate_rolleover_after_duration() (gas: 234428)
[PASS] test_setPaused_fail_non_admin() (gas: 15357)
[PASS] test_setRewardsDuration_after_period() (gas: 208313)
[PASS] test_setRewardsDuration_befere_period() (gas: 203628)
[PASS] test_setRewardsDuration_before_starting() (gas: 23909)
[PASS] test_setRewardsDuration_fail_non_admin() (gas: 15377)
[PASS] test_stake_fail_zero_amount() (gas: 36934)
[PASS] test_stake_increase_balance_check() (gas: 110574)
[PASS] test_withdraw_fail_zero_balance() (gas: 34794)
[PASS] test_withdraw_increase_LP_token_decrease_staking_token() (gas: 140461)
```

End of Document