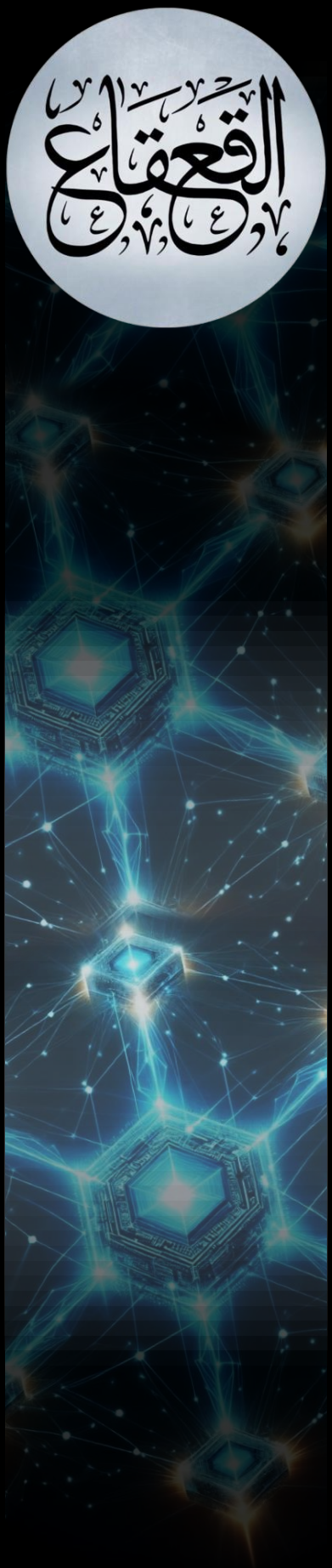
Al-Qa'qa’



DYAD

TimeLock Auditing Report

Auditor: Al-Qa'qa’

11 December 2024

Table of Contents

1 Introduction

1.1 About Al-Qa'qa'

Al-Qa'qa' is an independent Web3 security researcher who specializes in smart contract audits. Success in placing top 5 in multiple contests on code4rena and sherlock. In addition to smart contract audits, he has moderate experience in core EVM architecture, geth.

For security consulting, reach out to him on Twitter - @Al_Qa_qa

1.2 About DYAD

DYAD is a stablecoin protocol (Dollar pegged), it allows `ERC721` positions, where positions can be traded on third markets. In addition to this, they introduce the kerosine token, which its value depends on the volume of minted DYAD, and the locked collaterals, so it is as valuable as the degree of DYAD's over-collateralization.

1.3 Disclaimer

Security review cannot guarantee `100%` the safeness of the protocol, In the Auditing process, we try to get all possible issues, and we can not be sure if we missed something or not.

Al-Qa'qa' is not responsible for any misbehavior, bugs, or exploits affecting the audited
code or any part of the deployment phase.

And change to the code after the mitigation process, puts the protocol at risk, and should be audited again.

1.4 Risk Classification

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

1.4.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality
- Low - Funds are **not** at risk

1.4.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align or little-to-no incentive

2 Executive Summary

2.1 Overview

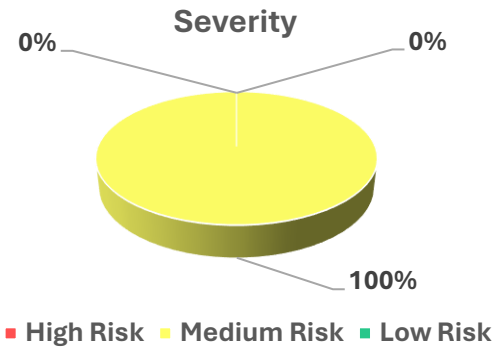
Project	**DYAD Stablecoin**
Repository	DyadStableCoin::xpv2, DyadStableCoin::feat/timelock
Commit Hash	84f8337d024ebf289e102f352fdb14b5fccc9418
Mitigation Hash	70f2a4bda83905af420b94a9d328a383a1c6e501
Audit Timeline	6 Nov 2024 to 8 Nov 2024

2.2 Scope

- script/deploy/Deploy.TimeLock.s.sol
- src/staking/DyadXPv2.sol

2.3 Issues Found

Severity	Count
High Risk	0
Medium Risk	2
Low Risk	0



3 Findings Summary

ID	Title	State
M-01	`onlyOwner` modifier prevents Note owner to update his Balance himself	Fixed
M-02	`setHalvingConfiguration()` is subjected to DoS risk	Acknowledged

4 Findings

4.1 Medium Risk

4.1.1 `onlyOwner` modifier prevents Note owner to update his Balance himself

context: DyadXPv2.sol#L125-L132

Description:
In the new version of DyadXP, there is a function that can be called to auto modify Note XP without performing depositing/withdrawing etc... from the VaultManager. This function is designed to be only called by Protoocl Admins or the owner of the Note, to update his balance. The problem is that Although the function is designed to be callable by Note owner, if Note Owner calls it it will revert because of the presence of `onlyOwner` modifier which will make the tx revert.

DyadXPv2.sol#L125-L132

```
    function forceUpdateXPBalance(uint256 noteId) external onlyOwner {
        if (msg.sender != owner()) {                          ^^^^^^^^^
            if (msg.sender != DNFT.ownerOf(noteId)) {
                revert Unauthorized();
            }
        }
        _updateNoteBalance(noteId);
    }
```

The function checks that the caller is the protocol owner; if not, it checks that the caller is the Note owner. But as `onlyOwner` modifier exists, calling it by Note owner will revert.

This will prevent Note Holders from updating their balance manually, which is a supported feature in the version of DyadXP.

Recommendations:
Remove `onlyOwner` modifier from the function.

```
diff --git a/src/staking/DyadXPv2.sol b/src/staking/DyadXPv2.sol
index c4b3dfc..17c9dfb 100644
--- a/src/staking/DyadXPv2.sol
+++ b/src/staking/DyadXPv2.sol
@@ -122,7 +122,7 @@ contract DyadXPv2 is IERC20, UUPSUpgradeable,
OwnableUpgradeable {
        _updateNoteBalance(noteId);
    }

-    function forceUpdateXPBalance(uint256 noteId) external onlyOwner {
+    function forceUpdateXPBalance(uint256 noteId) external {
        if (msg.sender != owner()) {
            if (msg.sender != DNFT.ownerOf(noteId)) {
                revert Unauthorized();
```

Sponser: Fixed at commit: 70f2a4bda83905af420b94a9d328a383a1c6e501
Al-Qa'qa': Verified. The issue has been fixed as recommended.

4.1.2 `setHalvingConfiguration()` is subjected to DoS risk

context: DyadXPv2.sol#L172-L179

Description:
When setting halving configurations, if there were halving configurations set before, we loop through all Notes and update their balance if they hold kerosine tokens.

DyadXPv2.sol#L172-L179

```
    function setHalvingConfiguration(uint40 _halvingStart, uint40 _halvingCadence)
external onlyOwner {
        ...
>>      if (halvingStart != 0) {
            uint256 dnftSupply = DNFT.totalSupply();
            for (uint256 i; i < dnftSupply; ++i) {
                if (noteData[i].accrualRate != 0) {
>>                  _updateNoteBalance(i);
                }
            }
        }
        ...
    }
```

The number of Notes holder kerosine tokens can be huge. There are currently `882` dNFT tokens, which means we will loop 882 times.

We will not call `_updateNoteBalance()` for all of them, as not all of them have accrualRate (kerosine deposited) greater than zero. But as time passes, more Notes will be minted, and more users will deposit kerosine, so the number of Loops iterations will increase, and the number of times we fire `_updateNoteBalance()` will increase, too.

At some point, we may not be able to set new halving configurations, as the cost of execution can exceed the Block gas limit.

This problem also existed in `initialize()`, so if the upgrading process is successful now as the number of Notes holding kerosine is small, upgrading the contract in the future may not be possible if the number of notes holding kerosine increases.

Recommendations:
Implementing a logic up updating halving configuration without making an iterative loop through all dNFTs is better.

Sponser: Acknowledged.