DYAD Security Review

Auditor: Al-Qa'qa’

30 August 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 overcollateralization.

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

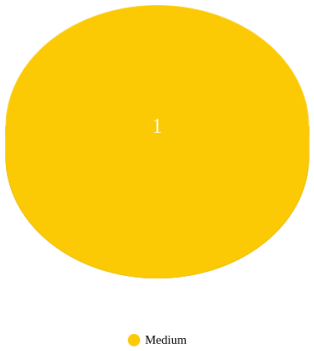
Project	DYAD Stablecoin
Repository	DyadStableCoin
Commit Hash	c230ef5b2cb6c3b6e60081f32d78b034a7a410cb
Mitigation Hash	f01baab8f9b8ccf18647c99d0f913d914fdad8fe
Audit Timeline	17 Aug 2024 to 18 Aug 2024

2.2 Scope

- src/core/Vault.weETH.sol
- script/deploy/Deploy.weETH.Vault.s.sol

2.3 Issues Found

Severity	Count
High Risk	0
Medium Risk	1
Low Risk	0
Total Issues	**1**



3 Finding Summary

ID	title	status
M-01	*depositCap* check is implemented incorrectly in *deposit()*	Resolved
I-01	Change parameters named to express the real meaning	Acknowledged

4 Findings

4.1 Medium Risk

4.1.1 *depositCap* check is implemented incorrectly in *deposit()*

Severity: MEDIUM

Context: Vault.weETH.sol#L57-L59

Description: When checking for *depositCap*, we should make the Vault Balance not exceeds this Cap, by checking the the Vault tokens + the deposited amount do not exceeds that Cap.

Vault.weETH.sol#L57-L59

```
  function deposit(uint id, uint amount) external onlyVaultManager {
❌      if (asset.balanceOf(address(this)) + amount > depositCap) {
        revert ExceedsDepositCap();
    }
  ...
  }
```

The problem here is that, when we are making a deposit, we call *VaultManger::deposit()* which sends the money to the vault before it calls *Vault::deposit()*

VaultManagerV4.sol#L95-L96

```
  function deposit( ... ) ... {
      ...
1:    _vault.asset().safeTransferFrom(msg.sender, vault, amount);
2:    _vault.deposit(id, amount);
      ...
  }
```

So this will end up making the cap check by adding the value two times instead of 1.

Proof of Concept:

- depositCap is *100*
- Vault Balance is *50*
- UserA deposited *30 eETH*
- transfer *30 eETH* to vault
- Vault Balance is 50 + 30 = *80*
- fire *vault.deposit()*
- Vault balance + amount > cap? is 80 + 30 > 100? true
- Revert the tx, and prevent the deposit

So as we can see the deposit will get prevented even it still didn't reached the depositCap.

Recommendations: Check for the Vault balance without adding the amount.

```
diff --git a/src/core/Vault.weETH.sol b/src/core/Vault.weETH.sol
index 5b1260a..45211ba 100644
--- a/src/core/Vault.weETH.sol
+++ b/src/core/Vault.weETH.sol
@@ -54,7 +54,7 @@ contract VaultWeETH is IVault, Owned {
    }

    function deposit(uint id, uint amount) external onlyVaultManager {
-       if (asset.balanceOf(address(this)) + amount > depositCap) {
+       if (asset.balanceOf(address(this)) > depositCap) {
            revert ExceedsDepositCap();
        }
        id2asset[id] += amount;
```

Sponsor: Fixed in PR-79, commit: c599e6029ccc03f78341ad1c677ea308c681850b.
Al-Qa'qa': Verified. The issue has been fixed as recommended.

4.2 Informational

4.2.1 Change parameters named to express the real meaning

Severity: INFO

Context:

- Deploy.weETH.Vault.s.sol#L21
- Deploy.weETH.Vault.s.sol#L24

Description: The VaultManager was a non-upgradable contract at first, then *DYAD* team made another version *v2*, and migrated to it, and they made it upgradable this time.

Now when doing upgrades there is no need to migrate things or something, we just change the implementation contract, and the VaultManager version is *v4* now.

The problem is that the naming of constant variants is still expressing the VaultManager by *v2*, we can see that in *Deploy.weETH.Vault.s.sol* how variables are named.

Deploy.weETH.Vault.s.sol#L19-L26

```
  new VaultWeETH(
    MAINNET_FEE_RECIPIENT,
@>    VaultManager (MAINNET_V2_VAULT_MANAGER),
    ERC20 (MAINNET_WEETH),
    IAggregatorV3(MAINNET_CHAINLINK_WEETH),
@>    IVault(MAINNET_V2_WETH_VAULT),
    DNft(MAINNET_DNFT)
  );
```

VaultManager is named *MAINNET_V2_VAULT_MANAGER*, and wethVault is named *MAINNET_V2_WETH_VAULT*, but the actual version is now *v4*.

Since the VaultManager is now a proxy contract, there is no need to name it with versions, to be easily understandable and improve code readability.

Recommendations: We can change the parameter name from *MAINNET_V2_VAULT_MANAGER* to *MAINNET_VAULT_MANAGER_PROXY*. and for *MAINNET_V2_WETH_VAULT* we can change it to either *MAINNET_WETH_VAULT*.

NOTE: we can add new variables in Parameters.sol beside the old variables instead of changing them, to not break the other deploying scripts

Sponser: Acknowledged.