Degen smart contracts
Security Review

Review by:
Milo Truck, Security Researcher

August 2, 2024

Contents

1 Introduction

1.1 Disclaimer

A security review a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While the review endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that a security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.2 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.2.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2   Security Review Summary

DEGEN is a reward token for users of Farcaster, utilizing a unique tip allowance mechanism that empowers the community to distribute funds and reward quality content creators. It also powers the Degen chain, one of the L3s launched on Base, offering a platform for building and using Degen apps.

From Jul 16th to Jul 17th the security researchers conducted a review of mode-lock/src on commit hash e7d9e63b. A total of **3** issues in the following risk categories were identified:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 1
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 1

3 Findings

3.1 Medium Risk

3.1.1 `minDepositAmount` **check in** `deposit()` **doesn't include the user's locked balance**

Severity: Medium Risk

Context: DegenLockToken.sol#L107-L110

Description: The `deposit()` function ensures that the `amount` currently being deposited is not less than `minDepositAmount`:

```
function deposit(uint256 amount) external nonReentrant {
    if (amount < minDepositAmount) {
        revert MinimumDepositNotMet();
    }
```

However, this check does not include the amount of DEGEN tokens currently locked by the user. If a user currently holds some LDEGEN and wants to lock more, the new amount of tokens to deposit has to be more than `minDepositAmount`. For example:

- `minDepositAmount` is 10,000 DEGEN.

- User calls `deposit()` with 10,000 DEGEN.

- After a period of time, the user wants to increase his locked DEGEN amount to 15,000.

- He calls `deposit()` with `amount = 5000e18` to increase his locked amount:

    - The check shown above reverts as `amount < minDepositAmount`.

As shown above, the user is forced to deposit more than `minDepositAmount` of DEGEN again, even though his current LDEGEN balance is above the minimum deposit.

Recommendation: Include the caller's LDEGEN balance in the check:

```
  function deposit(uint256 amount) external nonReentrant {
-     if (amount < minDepositAmount) {
+     if (balanceOf(msg.sender) + amount < minDepositAmount) {
          revert MinimumDepositNotMet();
      }
```

Degen: Fixed in commit ebaeaa35.

Reviewer: Fixed.

3.2 Low Risk

3.2.1 Multiple deposits will reset a user's lock duration

Severity: Low Risk

Context: DegenLockToken.sol#L115, DegenLockToken.sol#L128-L130

Description: Whenever a user calls `deposit()` to deposit some DEGEN, his deposit timestamp is set to `block.timestamp`:

```
depositTimestamps[msg.sender] = block.timestamp;
```

The user can only withdraw when `lockDuration` has passed after his deposit timestamp:

```
if (block.timestamp <= depositTimestamps[msg.sender] + lockDuration) {
    revert LockPeriodOngoing();
}
```

As such, if a user performs multiple deposits, the amount of time that his DEGEN tokens are locked will be reset to the latest deposit, even if his previous deposit was unlocked. For example:

- Assume `lockDuration` is 90 days.

- User calls `deposit()` with 15,000 DEGEN.

- 90 days passes, so his deposit is unlocked.

- User calls `deposit()` again with 10,000 DEGEN to increase his locked amount.

- Now, his 25,000 DEGEN is locked for the next 90 days.

Users might assume that the 15,000 DEGEN from the first deposit should remain unlocked, which is not the case.

Recommendation: Consider documenting that a new deposit of any amount will cause all LDEGEN held by the user to be locked for the next 90 days.

Degen: Documented in commit 9c421f27.

Reviewer: Fixed.

3.3 Informational

3.3.1 `updateMinDepositAmount()` **can increase** `minDepositAmount` **above a user's LDEGEN balance**

Severity: Informational

Context: DegenLockToken.sol#L107-L110, DegenLockToken.sol#L154-L159

Description: The `deposit()` function ensures that the `amount` currently being deposited is not less than `minDepositAmount`:

```
function deposit(uint256 amount) external nonReentrant {
    if (amount < minDepositAmount) {
        revert MinimumDepositNotMet();
    }
```

`minDepositAmount` can be increased by the owner through `updateMinDepositAmount()`:

```
function updateMinDepositAmount(
    uint256 newMinDepositAmount
) external onlyOwner {
    minDepositAmount = newMinDepositAmount;
    emit MinDepositAmountUpdated(minDepositAmount);
}
```

Therefore, if the owner increases `minDepositAmount`, it is possible for a user's locked DEGEN amount to be smaller than `minDepositAmount`:

- `minDepositAmount` is 10,000 DEGEN.

- User calls `deposit()` with 10,000 DEGEN.

- Owner calls `updateMinDepositAmount()` to increase `minDepositAmount` to 20,000 DEGEN.

- Now, the user's locked DEGEN amount is smaller than `minDepositAmount`.

Note that this has no impact as the user is still able to withdraw his locked DEGEN.

Recommendation: Consider documenting that increasing `minDepositAmount` might raise it above the LDEGEN balance of some users.

Degen: Documented in commit 5aa0b01d.

Reviewer: Fixed.