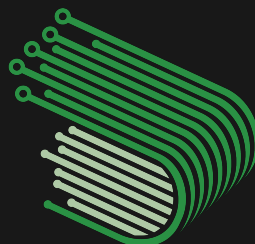


Security  
Audit

April 2024

# Goat.Tech

Audited by  
Ethan Bennett



**DARKLINEAR  
SOLUTIONS**

# Contents

<b>About Darklinear Solutions</b>	<b>I</b>
<b>Introduction</b>	<b>I</b>
<b>Finding</b>	<b>I</b>
Users can lock funds for less time than the minimum staking duration . . . . .	I

# About Darklinear Solutions

Darklinear Solutions provides unrivaled security for blockchain applications, from the bytecode to the browser. With years of experience in smart contract development and traditional software engineering, we find the bugs that others miss. Learn more at [darklinear.com](https://darklinear.com).

---

## Introduction

Goat.Tech is a gamified protocol for determining on-chain reputation. This review consists of an issue discovered during the course of Cantina's competitive audit for the Goat.Tech in April 2024. It does not represent a full and exhaustive audit of the protocol.

The finding described below is classified according to Cantina's standards.

---

## Finding

*Unique: No other researcher identified this issue*

### Users can lock funds for less time than the minimum staking duration

**Severity:** Medium risk

**Description:** Despite enforcing an explicit minimum staking duration, it is possible for users to lock funds for less time than intended.

If a user locks funds in an existing position without adding to its duration, the `Controller` checks (in `calMintStakingPower`) that the lock has not expired:

```
// LHelper.solL34

uint rd = LLocker.restDuration(oldLockData);
// ...
if (lockTime_ == 0) {
    require(rd > 0, "already unlocked");
}
```

It then allows staking to continue without additional checks on the `duration`. But if `rd` is less than the minimum staking duration, the new funds will only be locked for that arbitrarily short period of time. In contrast, the minimum staking amount is enforced in all cases.

**Recommendation:** Rather than requiring that the remaining duration of the position is greater than zero, the function should check that it is greater than the minimum staking duration:

```
// LHelper.solL23

function calMintStakingPower(
    LLocker.SLock memory oldLockData,
    uint lockAmount_,
    uint lockTime_,
    bool isSelfStake_,
    uint selfStakeAdvantage_,
    uint minDuration_
)
    internal
    view
    returns(uint)
{
    uint rd = LLocker.restDuration(oldLockData);
    // ...
    if (lockTime_ == 0) {
        require(
            rd > minDuration_,
            "too little time remaining"
        ); // minDuration_ = 30 days
    }
    // ...
}
```