TELLER
Security Review

DECEMBER, 2024

www.chaindefenders.xyz
https://x.com/ChDefendersEth

Lead Auditors

PeterSR

0x539.eth

Table of Contents

Protocol Summary

Teller is an extensible lending protocol for OTC loans. Lender groups is a contract stack on top that enables pool-style lending using the OTC loan backend, making for a unique permissionless architecture. (Can lend assets even if not allowlisted by our protocol)

Disclaimer

The ChainDefenders team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

Likelihood/Impact	High	Medium	Low
High	H	H/M	M
Medium	H/M	M	M/L
Low	M	M/L	L

Audit Details

Scope

Id	Files in scope
1	CollateralManager.sol
2	SmartCommitmentForwarder.sol
3	LenderCommitmentGroupShares.sol
4	LenderCommitmentGroup_Smart.sol
5	MarketRegistry.sol
6	TellerV2.sol
7	TellerV2Context.sol
8	TellerV2Storage.sol
9	CollateralEscrowV1.sol
10	UniswapPricingLibrary.sol
11	V2Calculations.sol

Roles

Id	Roles
1	ProtocolOwner
2	ProtocolFeeRecipient
3	PauserRole
4	User

Executive Summary

Issues found

Severity	Count	Description
High	0	Critical vulnerabilities
Medium	0	Significant risks
Low	1	Minor issues with low impact
Informational	2	Best practices or suggestions
Gas	2	Optimization opportunities

Findings

Medium

Mid 01 Anyone can remove lender and borrowers from MarketRegistry

Summary

In the current implementation of the `MarketRegistry` contract, any actor can revoke a lender or borrower for markets that require lender or borrower attestation.

Root Cause

The contract defines two versions of the `revokeLender` function:

1. Function `revokeLender(uint256 _marketId, address _lenderAddress, uint8 _v, bytes32 _r, bytes32 _s)`

   This function internally calls `_revokeStakeholderViaDelegation` but does not verify the signature (`v, r, s`) against the market owner, nor does it enforce any other ownership or permission checks. This omission allows unauthorized users to revoke stakeholders.

   Example code from `_revokeStakeholderViaDelegation`:

```
function _revokeStakeholderViaDelegation(
        uint256 _marketId,
        address _stakeholderAddress,
        bool _isLender,
        uint8 _v,
        bytes32 _r,
        bytes32 _s
    ) internal {
        bytes32 uuid = _revokeStakeholderVerification(
            _marketId,
            _stakeholderAddress,
            _isLender
        );
        // Note: Call to revoke attestation on EAS contracts is
    disabled.
    }
```

2. Function `revokeLender(uint256 _marketId, address _lenderAddress)`

   This version includes a check to ensure the caller is the market owner.

The discrepancy between these two versions creates an exploitable vulnerability.

Impact

Unauthorized revocation of lenders or borrowers can disrupt the proper functioning of markets that rely on stakeholder attestations. This could lead to:

- Denial of service for legitimate lenders and borrowers.
- Loss of trust in the platform.
- Potential financial losses for affected parties.

PoC

This vulnerability can be exploited by crafting a transaction to call revokeLender or revokeBorrower with arbitrary v, r, s values.

```
revokeLender(marketId, victimAddress, randomV, randomR, randomS);
```

This call bypasses ownership and attestation verification, removing the specified lender.

Mitigation

Add Verification : Update the revokeLender and revokeBorrower functions that accept v, r, s to enforce attestation verification against the market owner's signature