Accretion

# SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for MRGN
June 19, 2025
A25MRG1

MRGN

## AUDITORS

| Role | Name |
|---|---|
| Lead Auditor | Robert Reith (robert@accretion.xyz) |

## CLIENT

**MRGN** (https://marginfi.com) is one of the largest lending protocols in the Solana ecosystem. They engaged Accretion Labs to conduct a security assessment of a pull request for Marginfi, implementing a new feature called EMode. This feature allows the asset weight of correlated assets to be adjusted to more favorable terms. This way, when a user deploys mSOL as collateral, they will be able to borrow a correlated asset like SOL not at the default weight of maybe 50%, but instead a higher weight like 90% can be applied, because the asset prices are expected to move together. This leads to higher capital efficiency.

## ENGAGEMENT SCOPE

### Marginfi - EMode

**Link:** https://github.com/mrgnlabs/marginfi-v2/pull/318
**Commit:** 0a49436f682eac10ca94067a23f27d069c442583
**ProgramID:** MFv2hWf31Z9kbCa1snEPYctwafyhdvnV7FZnsebVacA

## ENGAGEMENT TIMELINE

14 Apr — Project Kickoff
Initial planning and scope definition

14 Apr — Assessment Begins
Security review and testing phase

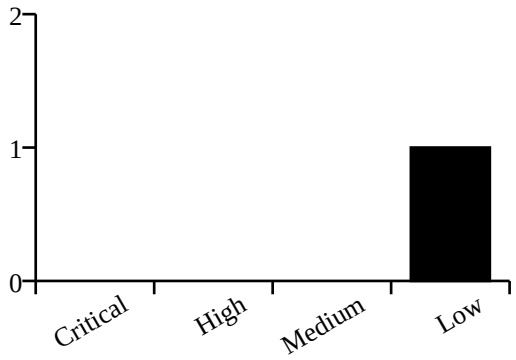
21 Apr — Review Fixes
Security recommendations are given and implemented

22 Apr — Project Completion
Report delivery and on-chain confirmation

## ASSESSMENT

The security assessment of MRGN's EMode implementation revealed that the new feature was built with much thought and integrated seamlessly into the existing product with no significant issues whatsoever. The only remark we had was about the possibility of configuring asset weights way beyond 100%, which would require a malicious or negligent admin. They quickly implemented an upper bound which also allows weights above 100% in case of emergency situations where this could be useful to prevent liquidations. Their team was already well aware of other potential issues with adjusting weights down leading to potential liquidations. Overall their product and this update are well built and maintained.

## SEVERITY DISTRIBUTION




## AUDITED CODE

### Program 1

**ProgramID:** MFv2hWf31Z9kbCa1snEPYctwafyhdvnV7FZnsebVacA
**Repository:** https://github.com/mrgnlabs/marginfi-v2
**Commit:** 1208ee17527087ac6eb480e64c51650eda24796f

## *ISSUES SUMMARY*

| ID | TITLE | SEVERITY | STATUS |
|---|---|---|---|
| ACC-L1 | Weights above one allowed for emode maintanace weights | low | fixed |

## *DETAILED ISSUES*

| **ID** | ACC-L1 |
|---|---|
| Title | Weights above one allowed for emode maintanace weights |
| Severity | low |
| Status | fixed |

### *Description*

The `validate_entries` function in `EmodeSettings` is used to confirm sane values for each emode entry. As each emode entry has both an `asset_weight_init` and an `asset_weight_maint` value, they are both validated. The former has to be between zero and one, while the latter's validation is limited to checking that it is larger or equal than the former. This allows weights way above one to be applied, either maliciously or by mistake. When a large maintenance weight is applied this could negatively affect the health of the overall lending protocol.

### *Location*

https://github.com/mrgnlabs/marginfi-v2/blob/0a49436f682eac10ca94067a23f27d069c442583/programs/marginfi/src/state/emode.rs#L99-L118

### *Relevant Code*

```
/// emode.rs L99-L118
    pub fn validate_entries(&self) -> MarginfiResult {
        for entry in self.emode_config.entries {
            if entry.is_empty() {
                continue;
            }
            let asset_init_w: I80F48 = I80F48::from(entry.asset_weight_init);
            let asset_maint_w: I80F48 = I80F48::from(entry.asset_weight_maint);

            check!(
                asset_init_w >= I80F48::ZERO && asset_init_w <= I80F48::ONE,
                MarginfiError::BadEmodeConfig
            );
            check!(asset_maint_w >= asset_init_w, MarginfiError::BadEmodeConfig);
        }

        // Validate that no duplicates exist (other than EMODE_TAG_EMPTY - 0)
        self.check_dupes()?;

        Ok(())
    }
```

### *Mitigation Suggestion*

Add an upper bounds check for the maintenance weight. It doesn't have to be one, but the possibility of setting a weight of 100 should probably be not supported.

### *Remediation*

Fixed in commit `1208ee17527087ac6eb480e64c51650eda24796f`.

## *APPENDIX*

### *Vulnerability Classification*

We rate our issues according to the following scale. Informational issues are reported informally to the developers and are not included in this report.

| Severity | Description |
|---|---|
| **Critical** | Vulnerabilities that can be easily exploited and result in loss of user funds, or directly violate the protocol's integrity. Immediate action is required. |
| **High** | Vulnerabilities that can lead to loss of user funds under non-trivial preconditions, loss of fees, or permanent denial of service that requires a program upgrade. These issues require attention and should be resolved in the short term. |
| **Medium** | Vulnerabilities that may be more difficult to exploit but could still lead to some compromise of the system's functionality. For example, partial denial of service attacks, or such attacks that do not require a program upgrade to resolve, but may require manual intervention. These issues should be addressed as part of the normal development cycle. |
| **Low** | Vulnerabilities that have a minimal impact on the system's operations and can be fixed over time. These issues may include inconsistencies in state, or require such high capital investments that they are not exploitable profitably. |
| **Informational** | Findings that do not pose an immediate risk but could affect the system's efficiency, maintainability, or best practices. |

### *Audit Methodology*

Accretion is a boutique security auditor specializing in Solana's ecosystem. We employ a customized approach for each client, strategically allocating our resources to maximize code review effectiveness. Our auditors dedicate substantial time to developing a comprehensive understanding of each program under review, examining design decisions, expected and edge-case behaviors, invariants, optimizations, and data structures, while meticulously verifying mathematical correctness—all within the context of the developers' intentions.

Our audit scope extends beyond on-chain components to include associated infrastructure, such as user interfaces and supporting systems. Every audit encompasses both a holistic protocol design review and detailed line-by-line code analysis.

During our assessment, we focus on identifying:
• Solana-specific vulnerabilities
• Access control issues
• Arithmetic errors and precision loss
• Race conditions and MEV opportunities
• Logic errors and edge cases
• Performance optimization opportunities
• Invariant violations
• Account confusion vulnerabilities
• Authority check omissions
• Token22 implementation risks and SPL-related pitfalls
• Deviations from best practices

Our approach transcends conventional vulnerability classifications. We continuously conduct ecosystem-wide security research to identify and mitigate emerging threat vectors, ensuring our audits remain at the forefront of Solana security practices.