Security Assessment Report

Marginfi V2 PRs 437, 441 and 450

January 16, 2026

Summary

The Sec3 team was engaged to conduct a thorough security analysis of the Marginfi V2 PRs 437, 441 and 450.

The artifact of the audit was the source code of the following programs, excluding tests, in https://github.com/mrgnlabs/marginfi-v2.

The initial audit focused on the following versions and revealed 1 issues or questions.

#	Task	Type	Commit
P1	PR437: Reduce only bank worth for borrows	Solana	9660a60
P2	PR441: Emode additional rate guards	Solana	473dde6
P3	PR450: Admin transfer/freeze account	Solana	c645072 → 8978771

This report provides a detailed description of the findings and their respective resolutions.

Table of Contents

Result Overview

Issue	Impact	Status
PR437: REDUCE ONLY BANK WORTH FOR BORROWS		
No issues found		
PR441: EMODE ADDITIONAL RATE GUARDS		
[P2-L-01] Missing emode leverage check in `lending_pool_configure_bank`	Low	Resolved
PR450: ADMIN TRANSFER/FREEZE ACCOUNT		
No issues found		

Findings in Detail

PR441: EMODE ADDITIONAL RATE GUARDS

[P2-L-01] Missing emode leverage check in `lending_pool_configure_bank`

Identified in commit `4838e6e`.

The emode leverage limit is currently only validated when updating emode settings in `lending_pool_configure_bank_emode`.

```
/* programs/marginfi/src/instructions/marginfi_group/config_bank_emode.rs */
007 | pub fn lending_pool_configure_bank_emode(
011 | ) -> MarginfiResult {
029 |     bank.emode.validate_entries_with_liability_weights(
030 |         &bank.config,
031 |         group.emode_max_init_leverage,
032 |         group.emode_max_maint_leverage,
033 |     )?;
050 | }
```

However, since leverage is calculated using both the emode weights and the bank's default weights, this validation should also occur when updating the bank's base weights in `lending_pool_configure_bank`.

```
/* programs/marginfi/src/state/emode.rs */
076 | fn validate_entries_with_liability_weights(
081 | ) -> MarginfiResult {
082 |     let liab_init_w: I80F48 = bank_config.get_weight(
083 |         RequirementType::Initial,
084 |         marginfi_type_crate::types::BalanceSide::Liabilities,
085 |     );
086 |     let liab_maint_w: I80F48 = bank_config.get_weight(
087 |         RequirementType::Maintenance,
088 |         marginfi_type_crate::types::BalanceSide::Liabilities,
089 |     );
094 |     for entry in self.emode_config.entries {
            // @audit: max_leverage_init is calculated using both base weight and emode weight
105 |         let max_leverage_init = calculate_max_leverage(asset_init_w, liab_init_w)?;
111 |         let max_leverage_maint = calculate_max_leverage(asset_maint_w, liab_maint_w)?;
116 |     }
122 | }

/* programs/marginfi/src/instructions/marginfi_group/configure_bank.rs */
018 | pub fn lending_pool_configure_bank(
020 |     bank_config: BankConfigOpt,
021 | ) -> MarginfiResult {
025 |     if bank.get_flag(FREEZE_SETTINGS) {
040 |     } else {
042 |         bank.configure(&bank_config)?;
058 |     }
061 | }
```

```
/* programs/marginfi/src/state/bank.rs */
337 | fn configure(&mut self, config: &BankConfigOpt) -> MarginfiResult {
          // @audit: base weight is updated here.
338 |     set_if_some!(self.config.asset_weight_init, config.asset_weight_init);
339 |     set_if_some!(self.config.asset_weight_maint, config.asset_weight_maint);
340 |     set_if_some!(
341 |         self.config.liability_weight_init,
342 |         config.liability_weight_init
343 |     );
344 |     set_if_some!(
345 |         self.config.liability_weight_maint,
346 |         config.liability_weight_maint
347 |     );
407 | }
```

It is recommended to add a call to the `validate_entries_with_liability_weights` function within `lending_pool_configure_bank`.

Resolution

Fixed by commit `02bffe3`.

Appendix: Methodology and Scope of Work

Assisted by the Sec3 Scanner developed in-house, the manual audit particularly focused on the following work items:

- Check common security issues.
- Check program logic implementation against available design specifications.
- Check poor coding practices and unsafe behavior.
- The soundness of the economics design and algorithm is out of scope of this work

DISCLAIMER

The instance report ("Report") was prepared pursuant to an agreement between Coderrect Inc. d/b/a Sec3 (the "Company") and MRGN, Inc. (the "Client''). This Report solely includes the results of a technical assessment of a specific build and/or version of the Client's code specified in the Report ("Assessed Code") by the Company. The sole purpose of the Report is to provide the Client with the results of the technical assessment of the Assessed Code. The Report does not apply to any other version and/or build of the Assessed Code. Regardless of the contents of the Report, the Report does not (and should not be interpreted to) provide any warranty, representation or covenant that the Assessed Code: (i) is error and/or bug free, (ii) has no security vulnerabilities, and/or (iii) does not infringe any third-party rights. Moreover, the Report is not, and should not be considered, an endorsement by the Company of the Assessed Code and/or of the Client. Finally, the Report should not be considered investment advice or a recommendation to invest in the Assessed Code and/or the Client.

This Report is considered null and void if the Report (or any portion thereof) is altered in any manner.

ABOUT

The Sec3 audit team comprises a group of computer science professors, researchers, and industry veterans with extensive experience in smart contract security, program analysis, testing, and formal verification. We are also building automated security tools that incorporate static analysis, penetration testing, and formal verification.

At Sec3, we identify and eliminate security vulnerabilities through the most rigorous process and aided by the most advanced analysis tools.

For more information, check out our website and follow us on twitter.

Sec3™