Accretion

SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for MRGN
November 17, 2025
A25MRG3

MRGN

AUDITORS

Role	Name
Lead Auditor	Robert Reith (robert@accretion.xyz)
Auditor	Mahdi Rostami (mahdi@accretion.xyz)

CLIENT

MRGN (https://marginfi.com) is one of the largest lending protocols in the Solana ecosystem. They engaged Accretion to perform a security assessment of new integrations for Marginfi, adding support for Drift and Solend. Using this feature, users can use Drift and Solend deposits as collateral within Marginfi.

ENGAGEMENT TIMELINE

30 Aug	●	Project Kickoff Initial planning and scope definition
30 Aug	●	Assessment Begins Security review and testing phase
13 Sep	●	Review Fixes Security recommendations are given and implemented
17 Nov	●	Project Completion Report delivery and on-chain confirmation

AUDITED CODE

Program 1

ProgramID: MFv2hWf31Z9kbCa1snEPYctwafyhdvnV7FZnsebVacA

Repository: https://github.com/mrgnlabs/marginfi-v2-internal/pull/115

ASSESSMENT

The security assessment of Marginfi's new protocol integrations evaluated the implementation of Drift and Solend deposit support as collateral. As one of the largest lending protocols in the Solana ecosystem, Marginfi demonstrated a robust implementation, with the review identifying only four findings: two low severity and two informational issues.

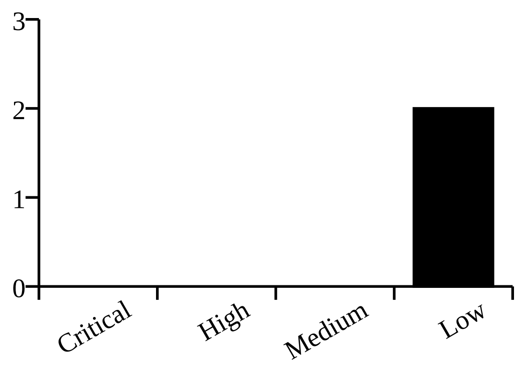
CODE ASSESSMENT

The code quality was consistently high across the new integrations. The implementation leverages the Anchor framework with coding patterns that integrate seamlessly with Marginfi's existing codebase. A comprehensive test suite was provided, demonstrating thorough coverage of the new Drift and Solend integration functionality.

KEY FINDINGS

Four issues were identified during the audit: two low severity and two informational findings. The low severity issues involved incorrect oracle handling in Drift deposit and withdraw operations for the USDC market, and missing `asset_tag` validation in `add_pool` instructions. The informational findings noted the use of Solend Obligation accounts as non-PDA accounts and missing Drift and Solend variants in the `is_oracle_error` function.

SEVERITY DISTRIBUTION



ENGAGEMENT SCOPE

The scope of this security assessment was a full review of the following items:

Item 1: Marginfi Drift and Solend Integration

Link: https://github.com/mrgnlabs/marginfi-v2-internal/pull/115
Commit: 03c8b76cbf561a77726bfc8908f69d5ebf7ec61b
Program ID: MFv2hWf31Z9kbCa1snEPYctwafyhdvnV7FZnsebVacA
Audit Result:

- **Audited Commit:** 1afaca0bed032ff2b73e76b650f79b06ccd4c608
- **Status:** Unverified
- **Comment:** Unverified

ISSUES SUMMARY

ID	TITLE	**SEVERITY**	**STATUS**
ACC-L1	Incorrect Oracle Handling in Drift Deposit and Withdraw for USDC Market	low	wontfix
ACC-L2	Missing `asset_tag` Validation in `add_pool` Instructions	low	fixed
ACC-I1	Using Solend Obligation as not a PDA	info	fixed
ACC-I2	`is_oracle_error` Missing Drift and Solend Variants	info	fixed

DETAILED ISSUES

ID	ACC-L1
Title	Incorrect Oracle Handling in Drift Deposit and Withdraw for USDC Market
Severity	low
Status	wontfix

Description

We found that during a **Drift funds deposit**, the instruction first calls `UpdateSpotMarketCumulativeInterest`. For the oracle account, if none is provided, it defaults to the **system program**.

The issue is that `handle_update_spot_market_cumulative_interest` requires the **exact correct oracle account** for validation. Passing the **system program** will fail validation, since the actual USDC spot market oracle is not the system account.

Validation: https://github.com/drift-labs/protocol-v2/blob/948fa5cb6524042d76bade8fc23fcd479cf669c9/programs/drift/src/instructions/keeper.rs#L2561-L2563

We checked the USDC spot market on-chain:

• Spot market account: `[[6gMq3mRCKf8aP3ttTyYhuijVZ2LGi14oDsBbkgubfLB3](https://solscan.io/account/6gMq3mRCKf8aP3ttTyYhuijVZ2LGi14oDsBbkgubfLB3#accountData)](https://solscan.io/account/6gMq3mRCKf8aP3ttTyYhuijVZ2LGi14oDsBbkgubfLB3#accountData)` • Oracle address: `9VCioxmni2gDLv11qufWzT3RDERhQE4iY5Gf7NTfYyAV` • Oracle owner: **Drift program** ([[link](https://solscan.io/account/9VCioxmni2gDLv11qufWzT3RDERhQE4iY5Gf7NTfYyAV#accountData)](https://solscan.io/account/9VCioxmni2gDLv11qufWzT3RDERhQE4iY5Gf7NTfYyAV#accountData))

This confirms that using the system account instead of the actual oracle is incorrect.

Location

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/drift/deposit.rs#L207-L211

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/drift/withdraw.rs#L322-L326

Relevant Code

```
oracle: self
            .drift_oracle
            .as_ref()
            .map(|o| o.to_account_info())
            .unwrap_or(self.system_program.to_account_info()), // USDC uses system program
```

Mitigation Suggestion

Update the logic to **pass the correct Drift oracle account** (e.g., `9VCioxmni2gDLv11qufWzT3RDERhQE4iY5Gf7NTfYyAV` for USDC) instead of defaulting to the system program.

Remediation

This issue has been marked as `wontfix` and comments have been added in commit `f844efd7a9a7a7e2081078105f887326a37c533d`.

ID	ACC-L2
Title	Missing `asset_tag` Validation in `add_pool` Instructions
Severity	low
Status	fixed

Description

In both `marginfi_group/add_pool_with_seed.rs` and `marginfi_group/add_pool.rs`, the `asset_tag` is not validated against `ASSET_TAG_DRIFT` and `ASSET_TAG_SOLEND`. Currently, the check only ensures it is not `ASSET_TAG_KAMINO`.

Although these instructions are permissioned (only the authority can call them), the lack of validation introduces inconsistency and could allow further issues.

Location

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/marginfi_group/add_pool_with_seed.rs#L52-L56

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/marginfi_group/add_pool.rs#L52-L56

Relevant Code

```
require_neq!(
        bank_config.asset_tag,
        ASSET_TAG_KAMINO,
        MarginfiError::CantAddPool
    );
```

Mitigation Suggestion

Add validation to also disallow `ASSET_TAG_SOLEND` and `ASSET_TAG_DRIFT`, in the same way `ASSET_TAG_KAMINO` is restricted.

Remediation

Fixed in commit `a73bcfbdf9dfdcbbe01141964f639c3bf9b788f1`.

ID	ACC-I1
Title	Using Solend Obligation as not a PDA
Severity	info
Status	fixed

Description

As I have seen before, the initialization of an obligation is typically permissionless, allowing anyone to perform this action after adding a pool. However, for Solend, the process is different due to the `solend_obligation`.

The `solend_obligation` must be owned by the Solend program during the initialization of the obligation, and when adding a pool, it is passed as a system account.

Therefore, before calling `init_obligation`, this account must be assigned to the Solend program. This creates a requirement for the current signer, disallowing users from initialize the Solend obligation before this step.

Another thing is that when a new Solend bank is created, the process involves two steps:

• add pool with a SystemAccount solend_obligation • init_obligation which initializes the solend_obligation through CPI However there is no enforcement that after a pool is added, the configured solend_obligation will be initialized through marginfi's init_obligation. Someone could add a pool with some solend_obligation, and then initialize the obligation through Solend directly, setting an external account as the obligation owner. Deposits and withdrawals may not work for that bank, but the owner of the obligation can still deposit/withdraw into it, and this just might mess with some accounting, even if its unusable through the protocol and shouldnt have any direct risk as i understand.

Location

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/solend/add_pool.rs#L177-L182

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/instructions/solend/init_obligation.rs#L80-L84

Relevant Code

```
    /// Obligation account for the marginfi program in Solend
    /// This will be initialized in a separate instruction
    /// TODO: Not sure the best way to handle this, it's randomly generated
    /// keypair and it should be available when calling add_pool. Just making
    /// sure it's not too much of an imposition
    pub solend_obligation: SystemAccount<'info>,
```

```
    /// The obligation account to be created. Note that the key was already generated when
    /// initializing the bank, and this must match the obligation recorded at that time.
    /// CHECK: validated by the Solend program that the account is empty and owned by solend program
    #[account(mut)]
    pub solend_obligation: UncheckedAccount<'info>,
```

Mitigation Suggestion

Turn Solend Obligation into a PDA.

Remediation

Fixed in commit `1afaca0bed032ff2b73e76b650f79b06ccd4c608`.

ID	ACC-I2
Title	`is_oracle_error` Missing Drift and Solend Variants
Severity	info
Status	fixed

Description

We found that the `is_oracle_error` helper does not account for `MarginfiError::DriftInvalidOracleSetup` or `MarginfiError::SolendInvalidOracleSetup`.

As a result, errors from Drift or Solend oracle setup are not classified as oracle errors, which could lead to inconsistent handling.

Location

https://github.com/mrgnlabs/marginfi-v2-internal/blob/03c8b76cbf561a77726bfc8908f69d5ebf7ec61b/programs/marginfi/src/errors.rs#L465-L490

Relevant Code

```
pub fn is_oracle_error(&self) -> bool {
        matches!(
            self,
            MarginfiError::WrongNumberOfOracleAccounts
                | MarginfiError::SwitchboardInvalidAccount
                | MarginfiError::PythPushInvalidAccount
                | MarginfiError::SwitchboardWrongAccountOwner
                | MarginfiError::PythPushFeedIdNonHexCharacter
                | MarginfiError::PythPushFeedIdMustBe32Bytes
                | MarginfiError::PythPushInsufficientVerificationLevel
                | MarginfiError::PythPushMismatchedFeedId
                | MarginfiError::StakedPythPushWrongAccountOwner
                | MarginfiError::PythPushWrongAccountOwner
                | MarginfiError::WrongOracleAccountKeys
                | MarginfiError::PythPushStalePrice
                | MarginfiError::SwitchboardStalePrice
                | MarginfiError::StakePoolValidationFailed
                | MarginfiError::InvalidBankAccount
                | MarginfiError::MissingBankAccount
                | MarginfiError::MissingPythAccount
                | MarginfiError::MissingPythOrBankAccount
                | MarginfiError::PythPushInvalidWindowSize
                | MarginfiError::OracleMaxConfidenceExceeded
                | MarginfiError::KaminoInvalidOracleSetup
        )
    }
```

Mitigation Suggestion

Extend `is_oracle_error` to also include:

• `MarginfiError::DriftInvalidOracleSetup` • `MarginfiError::SolendInvalidOracleSetup`

Remediation

Fixed in commit `83bdf08fa691f9d4496210787030f5abde30fd9c`.

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.