



SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for Beans Dot Fun
April 11, 2025
A25BEA1



Beans Dot Fun

AUDITORS

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

CLIENT

Beans Dot Fun (<https://beans.fun>) is a token launch platform for tokens associated with websites. Users can submit websites and bet on them. Once a day a website wins and a token gets launched. The platform serves as a discoverability and an attention measuring tool for websites.

ENGAGEMENT SCOPE

Beans Program

Link: <https://github.com/caddifi/normie-programs>

Commit: 1ba494b429e621b122e7fe52fc15622ef41c298a

ProgramID: 9etdkcoz5wnscduUW2DwxaMKYLFLruhiDFpAGgZ5SAJY

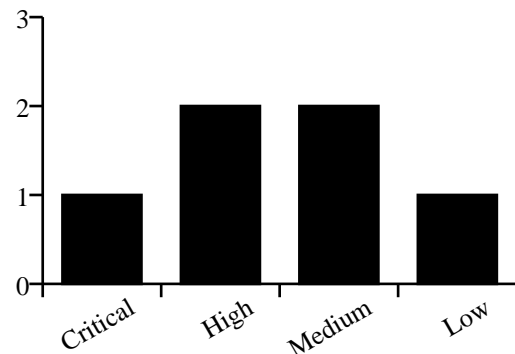
ENGAGEMENT TIMELINE

01 Mar	Project Kickoff Initial planning and scope definition
11 Mar	Assessment Begins Security review and testing phase
19 Mar	Review Fixes and updates Security recommendations are given and implemented
04 Apr	Project Completion Report delivery and on-chain confirmation

ASSESSMENT

The security assessment of the Beans token launch platform uncovered a few simple and serious vulnerabilities as the developer had newly adapted rust and solana development. This partially came from a complex architecture at the beginning of the audit. This architecture has been simplified later on, which has improved the overall security of the product. All issues have been fixed. The developer also implemented a thorough testing setup to verify the correctness of their onchain accounting.

SEVERITY DISTRIBUTION



AUDITED CODE

Program 1

ProgramID: 9etdkcoz5wnscduUW2DwxaMKYLFLruhiDFpAGgZ5SAJY

Repository: <https://github.com/caddifi/normie-programs>

Build Hash:

644c8d3ec5aaa17b7d99c90a58f44a49b2a9418c4587c88d08c31197f1a75574

Commit: d302cfff1126b47d15a453c4af354223906245ba

ISSUES SUMMARY

ID	TITLE	SEVERITY	STATUS
ACC-C1	Mint isn't associated with Domain	critical	fixed
ACC-H1	Buy winner beans sol not tracked in domain total bet	high	fixed
ACC-H2	Buy Winner doesn't check or change the claimed status of a bet entry	high	fixed
ACC-M1	Missing checks on supplied mint in create_mint	medium	fixed
ACC-M2	PreDeploy instruction can be DoSed	medium	fixed
ACC-L1	Claim can be DoSed	low	fixed
ACC-I1	Missing Sanity Checks in Update Global	info	fixed
ACC-I2	Missing Sanity Check in Create Global	info	fixed

DETAILED ISSUES

ID	ACC-C1
Title	Mint isn't associated with Domain
Severity	critical
Status	fixed

Description

When a mint is created using `create_mint`, this mint is not directly associated to the domain it belongs to. This means that later on, in `claim`, there is no way to confirm that the claim for a certain bet entry is using the correct mint. In fact there is no check in `claim` to confirm the mint address. Therefore, once multiple mints exist, the bet entry for one mint can be redeemed for a different mint, leading to potential critical Loss of Funds.

Location

https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/create_mint.rs#L60-L66 <https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/claim.rs#L39-L44>

Relevant Code

```
/// create_mint.rs L60-L66
#[account(
    init,
    payer = authority,
    mint::decimals = 9,
    mint::authority = global,
)]
pub mint: Box<Account<'info, TokenMint>>,

/// claim.rs L39-L44
#[account(
    mut,
    mint::decimals = 9,
    mint::authority = global,
)]
pub mint: Box<Account<'info, TokenMint>>,
```

Mitigation Suggestion

When creating a mint for a domain, that mint should be associated with that domain. This can be done multiple ways, for example by saving the mint Pubkey in the `Domain` account. Another way is to use a PDA as the mint address, and let this PDA use the domain account key as a seed.

Remediation

Fixed in commit `db186c87bf2dc9f9abfcff3fb2e2c153ce2770d5`.

ID	ACC-H1
----	--------

Title	Buy winner beans sol not tracked in domain total bet
-------	--

Severity	high
----------	------

Status	fixed
--------	-------

Description

When buy winner is used, the new domains `total_bet` amount is increased by the ``amount_non_beans_sol``, while the ``amount_beans_sol`` is not added to it, which creates a discrepancy in funds.

Location

https://github.com/caddifi/normie-programs/blob/9d8bdfd2fad72a30d73550d42a56dde02828f8f0/programs/beans/src/instructions/buy_winner.rs#L126-L129

Relevant Code

```
/// buy_winner.rs L126-L129
    bet_entry_next.total_bet += amount_non_beans_sol;
    bet_entry_next.total_bet_beans_sol += amount_beans_sol;
    bet_entry_next.total_fee += bet_entry_prev.total_fee;
    domain_next.total_bet += amount_non_beans_sol;
```

Mitigation Suggestion

Record the beans sol amount in the `domain_next` amount.

Remediation

The issue has been fixed in commit `a5fd28b94785f4acadeef2607f37b46e428397f0`.

ID	ACC-H2
Title	Buy Winner doesn't check or change the claimed status of a bet entry
Severity	high
Status	fixed

Description

The instruction `buy_winner` takes a bet entry from one domain, and uses its funds to buy another domain. When this is done, it doesn't check that the old ``bet_entry`` has already been claimed, and also doesn't invalidate an existing old ``bet_entry`` in any way. This means that this function can be called multiple times on the same ``bet_entry_prev``, it can be called on a claimed ``bet_entry``, and also users can still withdraw from a ``bet_entry`` after it has been used in ``buy_winner``.

Location

https://github.com/caddifi/normie-programs/blob/9d8bdfd2fad72a30d73550d42a56dde02828f8f0/programs/beans/src/instructions/buy_winner.rs#L79-L87

Relevant Code

```
/// buy_winner.rs L79-L87
pub fn buy_winner(&mut self) -> Result<()> {
    require!(
        self.bet_entry_prev.key() != self.bet_entry_next.key(),
        ErrorCode::InvalidBetEntry
    );
    require!(
        self.domain_next.status == Status::Live,
        ErrorCode::DomainLaunched
    );
}
```

Mitigation Suggestion

Check for claimed `bet_entries`, and also invalidated a ``bet_entry`` after this function is used.

Remediation

A check has been added to prevent a claimed `bet_entry` from being used in commit ``4e21eea13f001d0f1292626ab3af70e6183aa693``.

ID	ACC-M1
----	--------

Title	Missing checks on supplied mint in create_mint
-------	--

Severity	medium
----------	--------

Status	fixed
--------	-------

Description

The update in commit 34296d75 changed the `create_mint` function in such a way that the mint is not initialized, but supplied as an existing mint to the program. This means that the existing checks on the mint are insufficient. A malicious deployer could supply a mint that already has a significant chunk of supply minted, and also one where a freeze authority exists. However as the authority has to be the program authority the severity for this issue is lowered significantly.

Location

https://github.com/caddifi/normie-programs/blob/0007058b86a3c2580ccd0c546152b5bc3eb980c1/programs/beans/src/instructions/create_mint.rs#L59-L64

Relevant Code

```
/// create_mint.rs L59-L64
#[account(
    mut,
    mint::decimals = 9,
    mint::authority = global,
)]
pub mint: Box<Account<'info', TokenMint>>,
```

Mitigation Suggestion

Add additional checks on the mint: One that the freeze authority is set, and one that the supply is 0.

Remediation

Fixed in commit 7cae9cf824c108d1616c4055ad6b83c1cbef4818.

ID	ACC-M2
Title	PreDeploy instruction can be DoSed
Severity	medium
Status	fixed

Description

In the instruction `pre_deploy`, an Associated Token Account `global_mint_ata` and `beans_mint_ata` is initialized for authority ``global`` and ``authority`` respectively using the Anchor constraint ``init``. Because the pubkey of these authorities is predictable before this instruction is called, the Associated Token Account can be created by anyone before this instruction is called. Then, this instruction will fail because the account already exists and ``init`` expects a new account.

Location

https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/pre_deploy.rs#L48-L62

Relevant Code

```
/// pre_deploy.rs L48-L62
#[account(
    init,
    payer = authority,
    associated_token::mint = mint,
    associated_token::authority = global,
)]
pub global_mint_ata: Box<Account<'info', TokenAccount>>,

#[account(
    init,
    payer = authority,
    associated_token::mint = mint,
    associated_token::authority = authority,
)]
pub beans_mint_ata: Box<Account<'info', TokenAccount>>,
```

Mitigation Suggestion

Replace `init` with ``init_if_needed`` for the Associated Token Account.

Remediation

Fixed in commit `97f9931fca08ba51154f67a3e042eda10657c5b1`.

ID	ACC-L1
Title	Claim can be DoSed
Severity	low
Status	fixed

Description

In the instruction Claim, an Associated Token Account `user_mint_ata` is initialized for authority ``user`` using the Anchor constraint ``init``. Because the pubkey of this authority is predictable from the ``bet_entry`` before this instruction is called, the Associated Token Account can be created by anyone before this instruction is called. Then, this instruction will fail because the account already exists and ``init`` expects a new account. In this instance the severity is low because the user can simply close their own token account to make this instruction work.

Location

<https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/claim.rs#L46-L52>

Relevant Code

```
/// claim.rs L46-L52
#[account(
    init,
    payer = authority,
    associated_token::mint = mint,
    associated_token::authority = user,
)]
pub user_mint_ata: Box<Account<'info', TokenAccount>>,
```

Mitigation Suggestion

Replace `init` with ``init_if_needed`` for the Associated Token Account.

Remediation

Fixed in commit `97f9931fca08ba51154f67a3e042eda10657c5b1`.

ID	ACC-I1
Title	Missing Sanity Checks in Update Global
Severity	info
Status	fixed

Description

Multiple sanity checks that are done in `create_global` are missing in `update_global`. This includes `launch_time_period` and `vesting_period`. The checks should be the same in both functions.

Location

https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/update_global.rs#L26-L59

Relevant Code

```

/// update_global.rs L26-L59
pub fn update_global(
    &mut self,
    launch_time_period: Option<u64>,
    starting_discount: Option<u64>,
    vesting_period: Option<u64>,
    launch_idx: Option<u64>,
) -> Result<()> {
    let global = &mut self.global;
    if let Some(launch_time_period_val) = launch_time_period {
        debug!("Launch time period: {}", launch_time_period_val);
        global.launch_time_period = launch_time_period_val;
    }

    if let Some(starting_discount_val) = starting_discount {
        require!(
            starting_discount_val < 4615,
            ErrorCode::InvalidStartingDiscount
        );
        debug!("Starting discount: {}", starting_discount_val);
        global.starting_discount = starting_discount_val;
    }

    if let Some(vesting_period_val) = vesting_period {
        debug!("Vesting period: {}", vesting_period_val);
        global.vesting_period = vesting_period_val;
    }

    if let Some(launch_idx_val) = launch_idx {
        debug!("Launch idx: {}", launch_idx_val);
        global.launch_idx = launch_idx_val;
    }

    Ok(())
}

```

Mitigation Suggestion

Add the missing checks from `create_global` to `update_global`.

Remediation

Fixed in commit 97f9931fca08ba51154f67a3e042eda10657c5b1.

ID	ACC-I2
Title	Missing Sanity Check in Create Global
Severity	info
Status	fixed

Description

When global is initialized, the value for `starting_discount` is not checked. It is checked in ``update_global``, but not in ``create_global``.

Location

https://github.com/caddifi/normie-programs/blob/1ba494b429e621b122e7fe52fc15622ef41c298a/programs/beans/src/instructions/create_global.rs#L27-L48

Relevant Code

```
/// create_global.rs L27-L48
impl<'info> CreateGlobal<'info> {
    pub fn create_global(
        &mut self,
        bump: u8,
        launch_time_period: u64,
        starting_discount: u64,
        vesting_period: u64,
    ) -> Result<()> {
        let global = &mut self.global;
        require!(launch_time_period > 0, ErrorCode::InvalidLaunchTimePeriod);
        require!(
            launch_time_period < 60 * 60 * 24,
            ErrorCode::InvalidLaunchTimePeriod
        );
        require!(vesting_period > 0, ErrorCode::InvalidVestingPeriod);
        // require!(starting_discount < 4615, ErrorCode::InvalidStartingDiscount);
        global.bump = bump;
        global.claimable_sol = 0;
        global.launch_time_period = launch_time_period;
        global.launch_idx = 0;
        global.round_end_time = 1;
        global.starting_discount = starting_discount;
```

Mitigation Suggestion

Add a check in `create_global` that ``starting_discount_val < 4615`` to have the same condition as in ``update_global``

Remediation

Fixed in commit `97f9931fca08ba51154f67a3e042eda10657c5b1`.

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.