RISK AUDIT

for

Beanz

on

February 25, 2025

FIDESIUM

Executive Summary

Report

52
/100

TOTAL

Medium risk

February 25, 2025

Abstract

Fidesium's automated risk assessment service was requested to perform a risk posture audit on Block Asset **contracts**

Repository Link: https://github.com/caddifi/normie-programs

Initial Commit Hash:

`e42ddbae49e3388ec2f9d724bb320ac25c9fbb46`

Issue Summary

Critical	High	Medium	Low	Info
4 Issues	6 Issues	5 Issues	7 Issues	1 Issues

Caveats

Block Asset's codebase is well written, but does incur a handful of high value flaws.

Test Approach

Fidesium performed both Whitebox and Blackbox testing, as per the scope of the engagement, and relied on automated security testing.

Methodology

The assessment methodology covered a range of phases and employed various tools, including but not limited to the following:

- Mapping Content and Functionality of API
- Application Logic Flaws
- Access Handling
- Authentication/Authorization Flaws
- Brute Force Attempt
- Input Handling
- Source Code Review
- Fuzzing of all input parameter
- Dependency Analysis

Severity Definitions

Critical	The issue can cause large economic losses, large-scale data disorder or loss of control of authority management.
High	The issue puts users' sensitive information at risk or is likely to lead to catastrophic financial implications.
Medium	The issue puts a subset of users' sensitive information at risk, reputation damage or moderate financial impact.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is low-impact to the client's business.
Informational	The issue does not pose an immediate risk but is relevant to security best practices or defence in Depth.

Risk Issues

Vulnerability	Description	Risk	Probability	Status
Denial of Service	The `create_global.rs` script relies on `unwrap`.	Critical	Medium	Active
Frontrunning vulnerability	`bet.rs` relies on `init_if_needed` without proper access controls	Critical	Medium	Active
Reliance on build flags	`mainnet_constraint` macro bypasses all validation based on feature flag	Critical	Medium	Active
Static Single Component Seed	The `create_global.rs` script creates the PDA using a constant seed derivation (`global`).	Critical	Low	Active
Unchecked Metadata Creation	`create_mint.rs` does not validate metadata creation	High	Medium	Active
Missing authorization	`claim.rs` does not verify a users authority matches signer	High	High	Active
Missing authorization	`withdraw.rs` does not verify a users authority matches signer	High	High	Active
Missing account constraints	`init_uers.rs` does not validate `referrer_wallet` creation	High	Medium	Active
Missing init_if_needed validations	`buy_winner.rs` is missing crucial `init_if_needed` validations	High	Medium	Active
Direct lamport manipulation with `try_borrow_mut_lamports`	`bet.rs` uses `try_borrow_mut_lamports`	High	Medium	Active
Unchecked token supply	`claim.rs`does not validate token supply	Medium	Low	Active
Missing Signer Verification	`collect_fee.rs` does not validate the signer	Medium	Low	Active
Missing domain_bump validation	`bet.rs` does not validate domain_bump	Medium	Low	Active
Blacklist approach to state validation	`buy_winner.rs` validates `domain_next.status` against a blacklist	Medium	Low	Active
Missing Authority/Admin	`bet.rs`does not set authority or admin during `domain`initialization	Medium	Low	Active
Sybil vulnerability	`create_mint.rs` allows repeated invocation	Low	Medium	Active
Case sensitive url	`bet.rs` does not enforce consistent cases on URLs	Low	Low	Active
Reliance on Clock time	Multiple contracts rely on Clock time `Clock::get()`.	Low	Low	Active
Missing zero value validation	`withdra.rs` does not enforce non zero withdrawals	Low	Low	Active
Missing input validations	`create_global.rs` does not conduct sufficient input validation	Low	Low	Active
Missing URL validation	`bet.rs` does not validate url length, structure, or validity	Low	Low	Active
Reuse without Revalidation	`buy_winner.rs` reuses `total_fee` from `bet_entry_prev` without revalidation	Low	Low	Active
Unnecessary rent account	`init_users.rs` instantiates a dedicated rent account	Info	Info	Active

Risk Overview

Team Risk

Low risk: 1

No issues found in founding team

Doxxing Status	Team Experience	Risk Summary
Public	Highly relevant	Low

Liquidity

Risk summary: N/A

As this is a Github assessment, liquidity risks have not been assessed

Whale Concentration

Risk summary: N/A

As this is a Github assessment, whale risks have not been assessed

Smart Contract Risks

Risk summary: 57

The contracts are mostly well written, but have a handful of significant flaws that should to be carefuly managed.

Vulnerabilities Critical

Denial of Service

Vulnerability severity: **Critical**

Vulnerability probability: **Medium**

The `create_global.rs` script relies on `unwrap`.

An attacker could craft malformed data to appear valid and pass parsing. The `unwrap` call would then trigger a program panic.

Recommendations:

Replace the `unwrap` call with proper error handling

Frontrunning vulnerability

Vulnerability severity: **Critical**

Vulnerability probability: **Medium**

`bet.rs` relies on `init_if_needed` without proper access controls

An attacker could monitor the mempool and frontrunning initialization. Bets on the initialized domain would then affect potentially unexpected domains, leading to finacials loss

Recommendations:

- Ensure only verified domain owners can create domain accounts
- Prevent name squatting through verification
- Add explicit ownership records

Reliance on build flags

Vulnerability severity: **Critical**

Vulnerability probability: **Medium**

`mainnet_constraint` macro bypasses all validation based on feature flag

If a build pipeline errors or a developer misconfigures it, this could lead to total loss of control in production, given this is used to validate authority in `buy_winner.rs`

Additionally, this will make testing validation features challenging and unpredictable.

Recommendations:

Given this is only ever used for authority validation, Fidesium recommends

- Ensure only verified domain owners can create domain accounts
- Prevent name squatting through verification
- Add explicit ownership records

Vulnerabilities Critical

Static Single Component Seed

Vulnerability severity: **Critical**

Vulnerability severity: **Low**

The `create_global.rs` script creates the PDA using a constant seed derivation (`global`).

This leaves the program open to Account Injection Attack, Namespace Collision Attack, and Unauthorized access

Recommendations:

Make PDA derivation utilize program id, version/environment discrimniators, a namespace, and a dedicated type base seed, e.g:

```
...
#[account(
    init,
    payer = authority,
    space = 8 + Global::INIT_SPACE,
    seeds = [
        b"beans_protocol",
        b"global_config",
        b"v1",
        protocol_identifier.as_ref(),
        program_id.key().as_ref()
    ],
    bump,
)]
...
```

Vulnerabilities High

Unchecked Metadata Creation

Vulnerability severity: **High**

Vulnerability probability: **Medium**

`create_mint.rs` does not validate metadata creation

An attacker could craft malicious metadata, leading to token impersonation, metadata poisoning, storage exploitation, or market manipulation

Recommendations:

- Validate url input

```
require!(
    url.len() <= MAX_URL_LENGTH && is_valid_url_format(&url),
    ErrorCode::InvalidUrlFormat
    );
```

- Verify metadata

```
let metadata_account = Metadata::from_account_info(&self.metadata)?;
require!(
    metadata_account.mint == mint.key() &&
    metadata_account.update_authority == global.key(),
    ErrorCode::MetadataVerificationFailed
);
```

- Verify URL content safety

```
let metadata_uri = format!("https://beans.fun/token/{}", encoded_url);
require!(
    metadata_uri.len() <= MAX_URI_LENGTH,
    ErrorCode::UriTooLong
);
```

- Add separate post creation correctness validations

Vulnerabilities High

Missing authorization

Vulnerability severity: **High**

Vulnerability probability: **High**

`withdraw.rs` does not verify a users authority matches signer

An attacker could pass in an invalid user account or withdraw bets he does not control

Recommendations:

```
require!(
    bet_entry.user == authority.key(),
    ErrorCode::UnauthorizedWithdrawal
);
```

Missing authorization

Vulnerability severity: **High**

Vulnerability probability: **High**

`claim.rs` does not verify a users authority matches signer

An attacker could pass in an invalid user account

Recommendations:

```
#[account(
    mut,
    constraint = user.authority == authority.key() @ ErrorCode::UnauthorizedUser,
)]
pub user: Box>,
```

Missing account constraints

Vulnerability severity: **High**

Vulnerability probability: **Medium**

`init_users.rs` does not validate `referrer_wallet` creation

Recommendations:

```
#[account(
    constraint = referrer_wallet.is_some() && referrer_pda.is_some()
    && referrer_pda.as_ref().unwrap().authority == referrer_wallet.as_ref().unwrap().key()
)]
```

Vulnerabilities High

Missing init_if_needed validations

Vulnerability severity: **High**

Vulnerability probability: **Medium**

`buy_winner.rs` is missing crucial `init_if_needed` validations

- Ownerhsip Validation
- Content Validation
- User Authorization
- Initial State Constraints

Recommendations:

```
#[account(
    init_if_needed,
    payer = authority,
    space = 8 + BetEntry::INIT_SPACE,
    seeds = [b"bet_entry", url_to.as_bytes(), user_keypair.as_ref(), launch_idx.to_le_bytes().as_ref()],
    bump,
    constraint = !bet_entry_next.initialized ||
        (bet_entry_next.domain == domain_next.key() &&
         bet_entry_next.launch_idx == launch_idx),
    )]
```

Direct lamport manipulation with `try_borrow_mut_lamports`.

Vulnerability severity: **High**

Vulnerability probability: **Medium**

Multiple functions use `try_borrow_mut_lamports`

This bypasses the Account System, does not provide transaction atomicity guarantees, and alows for fund loss

- bet.rs
- buy_winner.rs
- withdraw.rs
- create_mint.rs
- collect_fee.rs

Recommendations:

Use Cross Program Invocation to the System Program with PDA signing for SOL transfers

Vulnerabilities Medium

Unchecked token supply

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`claim.rs`does not validate token supply

Recommendations:

- Add Explicit token supply validations
- Implement a distribution cap per domain
- Add mint authority controls
- Add global supply tracking

Missing Signer Verification

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`collect_fee.rs` does not validate the signer

Recommendations:

Add and verify an authority account.

Missing domain_bump validation.

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`bet.rs` does not validate domain_bump

This could lead to transaction failures, transaction highjacking, or even signature verification bypass

Recommendations:

Validate domain_bump:

```
    let (expected_domain_address, calculated_bump) = Pubkey::find_program_address(
        &[b"domain", url.as_bytes(), launch_idx.to_le_bytes().as_ref()],
        ctx.program_id
    );

    require!(
        domain_bump == calculated_bump && domain.key() == expected_domain_address,
        ErrorCode::InvalidBump
    );
```

Vulnerabilities Medium

Blacklist approach to state validation.

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`buy_winner.rs` validates `domain_next.status` against a blacklist

```
    require!(
        domain_next.status != Status::Launched,
        ErrorCode::DomainLaunched
    );
```

Blacklist validations are unreliable and lead to logic errors

Recommendations:

- Validate what `domain_next.status` is, instead of what it isn't
- Conduct a full state machine check

Missing Authority/Admin

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

This means there's no explicit ownership or control over who can modify the domain later.

An attacker could potentially initialize domains they shouldn't have access to

Recommendations:

- Validate authority

```
    require!(
        authority.key() == global.admin
```

- Assign an authority to `domain`

Vulnerabilities Low

Sybil vulnerability

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`create_mint.rs` allows repeated invocation

An attacker could repeatedly call the function, sybillin themselves, and overconsuming rent, while leading to chain storage bload

Recommendations:

Implement Rate Limiting, cost barriers, and a resource allocation cap

Case sensitive url

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`bet.rs` does not enforce consistent cases on URLs.

```
                    domain.url = url.clone()
```

Since URLs are case insensitive this could lead to URL clashes.

Recommendations:

Cast URLs to a consistent case, with e.g.:`to_lowercase()`

Reliance on Clock time

Vulnerability severity: **Low**

Vulnerability probability: **Low**

Multiple contracts rely on Clock time `Clock::get()`.

Clock time could be manipulated within a block, potentially leading to unexpected transaction orderings or other race conditions.

Recommendations:

- Use slot numbers in addition to clocktime to enforce ordering
- Implement buffer periods to avoid last second manipulations

Vulnerabilities Low

Missing zero value validation

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`withdra.rs` does not enforce non zero withdrawals

Recommendations:

```
require!(
    amount_non_beans_sol > 0 || amount_beans_sol > 0,
    ErrorCode::ZeroWithdrawalAmount
);
```

Missing input validations

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`create_global.rs` does not conduct sufficient input validation

- num_rounds lacks validation
- launch_time_period lacks validation
- vesting_period lacks validation

Recommendations:

Validate upper and lower bounds as well as non zero status

Missing URL validation

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`bet.rs` does not validate url length, structure, or validity

This could lead to storage bloat or malformed data.

Recommendations:

- Validate URL min and max length
- Validate URL structure
- Introduce a DNS verification Oracle, to enforce domain validity

Vulnerabilities Low

Reuse without Revalidation

Vulnerability severity: **Low**

Vulnerability probability: **Low**

`buy_winner.rs` reuses `total_fee` from `bet_entry_prev` without revalidation

An attacker could manipulate the prior `total_fee` potentially leading to loss of funds

This could lead to storage bloat or malformed data.

Recommendations:

Revalidate current state of `total_fee` before reuse

Vulnerabilities Info

Unnecessary rent account

Vulnerability severity: **Info**

Vulnerability probability: **Info**

`init_users.rs` instantiates a dedicated rent account

This is no longer necessary in modern Solana programs

Recommendations:

Remove rent account

Disclaimer

Disclaimer

This report is governed by the Fidesium terms and conditions.

This report does not constitute an endorsement or disapproval of any project or team, nor does it reflect the economic value or potential of any related product or asset. It is not investment advice and should not be used as the basis for investment decisions. Instead, this report provides an assessment intended to improve code quality and mitigate risks inherent in cryptographic tokens and blockchain technology.

Fidesium does not guarantee the absence of bugs or vulnerabilities in the technology assessed, nor does it comment on the business practices, models, or regulatory compliance of its creators. All services, reports, and materials are provided "as is" and "as available," without warranties of any kind, including but not limited to merchantability, fitness for a particular purpose, or non-infringement.

Cryptographic assets and blockchain technologies are novel and carry inherent technical risks, uncertainties, and the possibility of unpredictable outcomes. Assessment results may contain inaccuracies or depend on third-party systems, and reliance on them is solely at the Customer’s risk.

Fidesium assumes no liability for content inaccuracies, personal injuries, property damages, or losses related to the use of its services, reports, or materials. Third-party components are provided "as is," and any warranties are strictly between the Customer and the third-party provider.

These services and materials are intended solely for the Customer's use and benefit. No third party or their representatives may claim rights to or rely on these services, reports, or materials under any circumstances.