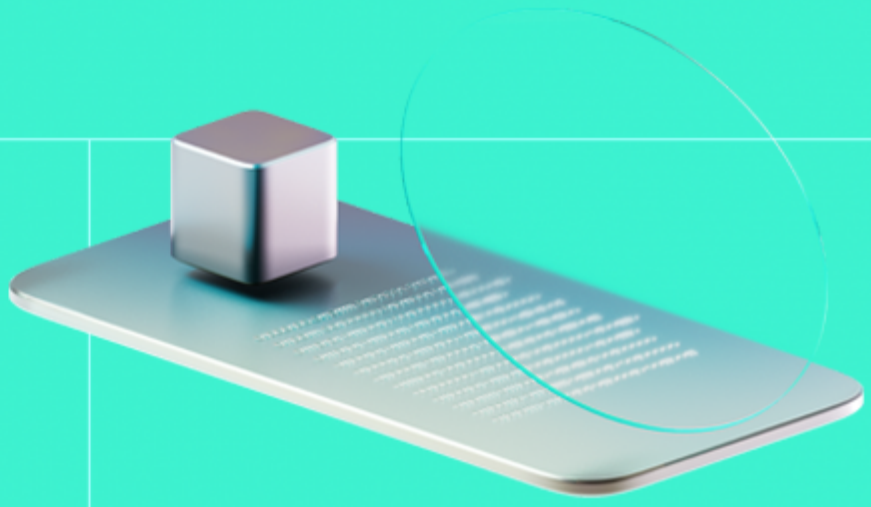
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Sociogram

Date: 17/02/2025

We express our gratitude to the Sociogram team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Sociogram is a token launch protocol on Solana that implements a bonding curve mechanism for predictable token distribution and price discovery. It provides a fully automated market maker (AMM) with virtual reserves for price calculation and real reserves for actual assets, allowing projects to launch tokens with guaranteed liquidity and controlled price movement. The protocol features a complete token lifecycle management system, from initial token creation and distribution through a bonding curve to final withdrawal phase when all tokens are sold.

Document

Name	Smart Contract Code Review and Security Analysis Report for Sociogram
Audited By	Jakub Heba
Approved By	Grzegorz Trawiński
Website	https://sociogram.org/
Changelog	21/01/2025 - Preliminary Report
	17/02/2025 - Final Report
Platform	Solana
Language	Rust
Tags	AMM, Curve
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/biswap-org/curve-launchpad
Commit	05f59a60823a6b60ce706ee9dffd94e2837ca1a2

Audit Summary

The system users should acknowledge all the risks summed up in the risks section of the report

3	2	1	0
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	0
Low	3

Vulnerability	Severity
F-2025-8330 - Lack of functionality for admin rotation	Low
F-2025-8331 - Missing functionality to pause program operations	Low
F-2025-8332 - Lack of validation for set_fee edge values	Low

Documentation quality

- Functional requirements are partially missed.
- Technical description is not provided.

Code quality

- Several template code patterns were found.
- The development environment is configured.

Test coverage

Code coverage of the project is N/A.

The Score is N/A since there is no reliable tool to calculate Solana test coverage

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is missed.
- Some of the tests were not working properly without adjustments.
- More advanced buy/sell cases are not tested thoroughly.

Table of Contents

System Overview

Sociogram is a bonding curve protocol on Solana that enables token launches through an automated market maker mechanism. The protocol consists of two core components:

- Bonding curve - manages token distribution through a customizable bonding curve mechanism, handling buy and sell operations with separate virtual and real reserve tracking for predictable price discovery.
- Global state - contains protocol configuration including fee parameters, reserve settings, and privileged addresses. It manages the overall protocol state and controls access to administrative functions.

Privileged roles

- The **authority** can modify protocol parameters including:
  - Fee recipient and fee basis points
  - Initial virtual and real token reserves
  - Withdraw authority address
- The **withdraw authority** has permission to withdraw remaining tokens and SOL from the bonding curve, but only after all tokens are sold and the curve reaches completion state. This role cannot interfere with active trading operations.

Potential Risks

- Absence of time-lock mechanisms for critical operations - Without time-locks on critical operations, there is no buffer to review or revert potentially harmful actions, increasing the risk of rapid exploitation and irreversible changes.
- Insufficient multi-signature controls for critical functions - The lack of multi-signature requirements for key operations centralizes decision-making power, increasing vulnerability to single points of failure or malicious insider actions, potentially leading to unauthorized transactions or configuration changes.
- Single entity upgrade authority - The token ecosystem grants a single entity the authority to implement upgrades or changes. This centralization of power risks unilateral decisions that may not align with the community or stakeholders' interests, undermining trust and security, for example changing the fee to the enormous high value.

Findings

Vulnerability Details

F-2025-8330 - Lack of functionality for admin rotation - Low

Description: The program does not provide a mechanism to rotate or update the authority. This is considered a deviation from leading security practices as it limits flexibility and poses operational risks, such as the inability to recover from admin key compromise.

```
pub fn initialize(ctx: Context<Initialize>) -> Result<()> {
    msg!("Calling initialize");
    let global = &mut ctx.accounts.global;

    require!(
        !global.initialized,
        CurveLaunchpadError::AlreadyInitialized,
    );

    global.authority = *ctx.accounts.authority.to_account_info().key;
```

Assets:

- programs/curve-launchpad/src/instructions/initialize.rs [https://github.com/biswap-org/curve-launchpad.git]

Status: Accepted

Classification

Impact: 1/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Simple

Severity: Low

Recommendations

Remediation: Implement a two-step authority transfer process to ensure a secure and verifiable handover of administrative privileges.

Step 1: Initiate Authority Transfer:

The current authority initiates the process by specifying the new authority's public key.

This step stores the new authority's public key but does not finalize the transfer.

Step 2: Confirm Authority Transfer:

The new authority confirms the transfer to finalize the process and update the program's admin key.

Resolution:

Issue was partially fixed in `0d67840978f3965c343007707618bc37306f4c2b`. The protocol implements a way for authority change, however the implemented approach is that old owner have to sign the authority changing transaction with the new owner.

This might be problematic, as a new owner have only few minutes for it, as Solana is restricting that timeframe to some specific blocks amount. If the new owner will be, for example, a multisig, such a change might require to be executed on one device, or via some automation.

F-2025-8331 - Missing functionality to pause program operations - Low

Description: The program lacks a mechanism to pause its operations. This is considered a bad practice as it limits the authority's ability to respond to critical situations, such as:

Security Incidents - in the event of a vulnerability or exploit, the inability to pause operations could lead to significant financial losses.

Operational Failures - Pausing functionality allows the authority to temporarily halt operations while addressing bugs, misconfigurations, or other unforeseen issues.

Assets:

- programs/curve-launchpad/instructions/set_params.rs [https://github.com/biswap-org/curve-launchpad.git]

Status: Fixed

Classification

Impact: 1/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Simple

Severity: Low

Recommendations

Remediation: Implement a pausability mechanism controlled by the authority. The program should maintain a state variable to indicate whether operations are paused, and all instructions should check this state before executing.

Resolution: Issue was fixed in `7d4bc557c2954f96b0723f5ea464774336710dd6` by creation of pausing and resuming mechanism.

F-2025-8332 - Lack of validation for set_fee edge values - Low

Description: The program does not validate the `fee_amount` value set during the `set_fee` call This allows authority to supply an arbitrary high values, what might discourage usage of the protocol due to the too big fee for the potential users. Additionally, authority can potentially front-run user operations, collecting high fees.

```
pub fn set_fee(
    ctx: Context<SetFee>,
    fee_amount: u64
) -> Result<()> {
    let global = &mut ctx.accounts.global;

    //confirm program is initialized
    require!(
        global.initialized,
        CurveLaunchpadError::NotInitialized
    );

    //confirm user is the authority
    require!(
        global.authority == *ctx.accounts.user.to_account_info().key,
        CurveLaunchpadError::InvalidAuthority
    );

    global.fee_basis_points = fee_amount;

    Ok(())
}
```

Assets:

- programs/curve-launchpad/src/instructions/set_fee.rs [https://github.com/biswap-org/curve-launchpad.git]

Status: Fixed

Classification

Impact: 1/5

Likelihood: 1/5

Exploitability: Independent

Complexity: Simple

Severity: Low

Recommendations

Remediation: Add validation logic to ensure that the `fee_amount` is between the acceptable bounds.

Resolution: Issue was fixed in the `8675be6574a8390696a02ab647de6c67e97013b6` commit by restricting the fee to be less or equal to 10%.

Observation Details

F-2025-8333 - Typo in the buy function - Info

Description: It was identified that one of the variables used within the `buy` function is `targe_token_amount`. This parameter has a typo in its name, as it should be correctly named `target_token_amount`.

```
    require!(token_amount > 0, CurveLaunchpadError::MinBuy,);

    let targe_token_amount = if ctx.accounts.bonding_curve_token_account.amou
nt < token_amount {
        ctx.accounts.bonding_curve_token_account.amount
    } else {
        token_amount
    };
```

While this is not a threat in itself, future code development may overlook this difference in names and mistakenly declare a new variable instead of overwriting or an existing one.

Assets:

- programs/curve-launchpad/src/instructions/buy.rs [https://github.com/biswap-org/curve-launchpad.git]

Status: Fixed

Recommendations

Remediation: We suggest adjusting the variable name to the correct one without the typo, that is `target_token_amount`.

Resolution: Issue was fixed in `e39b905415cc56a1c7f07e89fa712ef22cff3d12` by changing the variable name to `target_token_amount`.

F-2025-8334 - The MinSOLOutputExceeded error name is misleading - Info

Description: It was noted that if `sell_amount_minus_fee < min_sol_output`, an error named `CurveLaunchpadError::MinSOLOutputExceeded` is returned. This is misleading, since the statement "min output exceeded" is linguistically incorrect in this context.

```
//confirm min sol output is greater than sol output
require!(
    sell_amount_minus_fee >= min_sol_output,
    CurveLaunchpadError::MinSOLOutputExceeded,
);
```

The correct error should say that `sell_amount_minus_fee` is less than `min_sol_output`, for example using the `MinSOLOutputNotReached` error name.

Assets:

- programs/curve-launchpad/src/instructions/sell.rs [https://github.com/biswap-org/curve-launchpad.git]

Status: Fixed

Recommendations

Remediation: We suggest that the error name be adjusted to something more descriptive of the problem that occurred during the transaction invocation.

Resolution: Issue was fixed in `d2c90ecc4d20e3908c021e43f4dac865988a24ee` by changing the error name to `MinSOLOutputNotReached`.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Definitions

Severities

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution.

Potential Risks

The "Potential Risks" section identifies issues that are not direct security vulnerabilities but could still affect the project's performance, reliability, or user trust. These risks arise from design choices, architectural decisions, or operational practices that, while not immediately exploitable, may lead to problems under certain conditions. Additionally, potential risks can impact the quality of the audit itself, as they may involve external factors or components beyond the scope of the audit, leading to incomplete assessments or oversight of key areas. This section aims to provide a broader perspective on factors that could affect the project's long-term security, functionality, and the comprehensiveness of the audit findings.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details	
Repository	https://github.com/biswap-org/curve-launchpad
Commit	05f59a60823a6b60ce706ee9dffd94e2837ca1a2
Whitepaper	https://docs.biswap.org/biswap/core-products/biswap-v3/v3-whitepaper
Requirements	-
Technical Requirements	-

Asset	Type
programs/curve-launchpad/instructions/set_params.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/amm/amm.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/amm/mod.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/buy.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/constants.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/create.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/errors.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/events.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/initialize.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/mod.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/sell.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract
programs/curve-launchpad/src/instructions/set_fee.rs [https://github.com/biswap-org/curve-launchpad.git]	Smart Contract

Appendix 3. Additional Valuables

Additional Recommendations

The smart contracts in the scope of this audit could benefit from the introduction of automatic emergency actions for critical activities, such as unauthorized operations like ownership changes or proxy upgrades, as well as unexpected fund manipulations, including large withdrawals or minting events. Adding such mechanisms would enable the protocol to react automatically to unusual activity, ensuring that the contract remains secure and functions as intended.

To improve functionality, these emergency actions could be designed to trigger under specific conditions, such as:

- Detecting changes to ownership or critical permissions.
- Monitoring large or unexpected transactions and minting events.
- Pausing operations when irregularities are identified.

These enhancements would provide an added layer of security, making the contract more robust and better equipped to handle unexpected situations while maintaining smooth operations.