BLOCKAPEX

# SMART CONTRACT SECURITY

V 1.0

DATE: 06-03-2025

PREPARED FOR: PUMPKIN.FUN

## About BlockApex

Founded in early 2021 BlockApex is a security-first blockchain consulting firm. We offer services in a wide range of areas including Audits for Smart Contracts, Blockchain Protocols, Tokenomics along with Invariant development (i.e., test-suite) and Decentralized Application Penetration Testing. With a dedicated team of over 40+ experts dispersed globally, BlockApex has contributed to enhancing the security of essential software components utilized by many users worldwide, including vital systems and technologies.

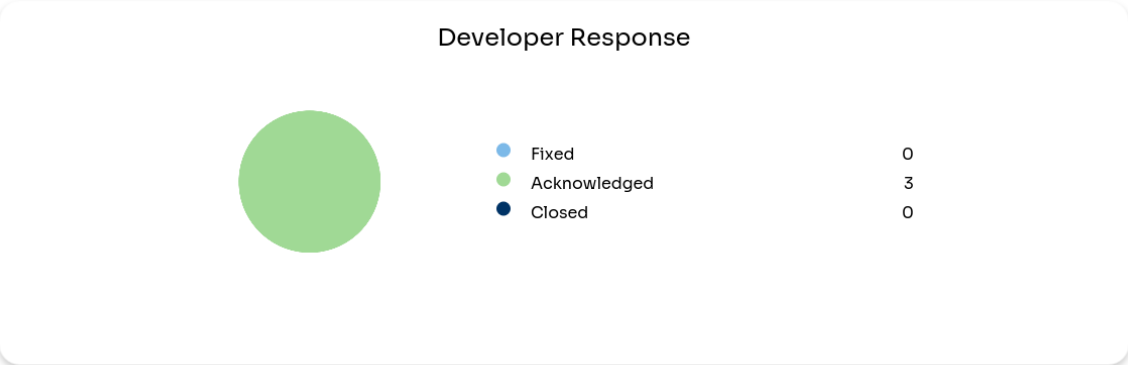
BlockApex has a focus on blockchain security, maintaining an expertise hub to navigate this dynamic field. We actively contribute to security research and openly share our findings with the community. Our work is available for review at our public repository, showcasing audit reports and insights into our innovative practices.

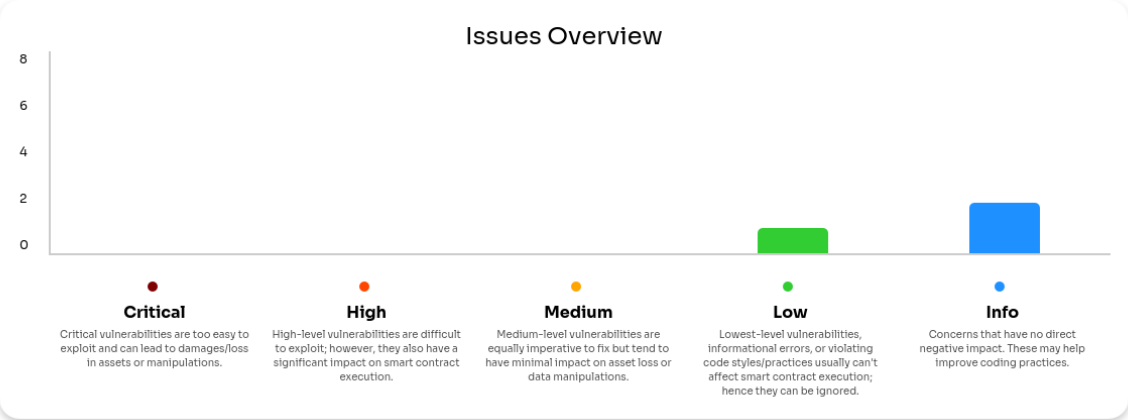
To stay informed about BlockApex's latest developments, breakthroughs, and services, we invite you to follow us on Twitter and explore our GitHub. For direct inquiries, partnership opportunities, or to learn more about how BlockApex can assist your organization in achieving its security objectives, please visit our Contact page at our website , or reach out to us via email at hello@blockapex.io.

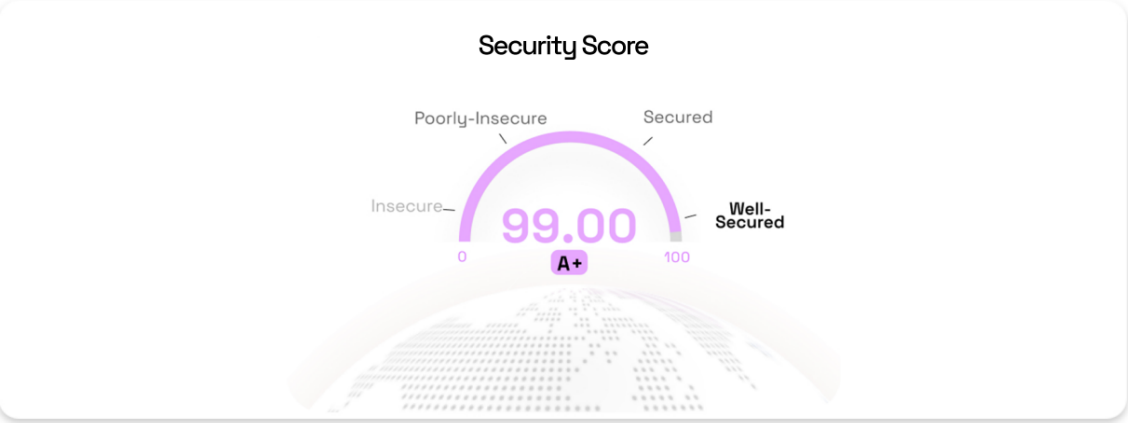
# Contents

# 1 Executive Summary

Our team conducted a Filtered Audit, engaging three auditors to independently examine the Pumpkin.fun Programs. This rigorous approach included a meticulous line-by-line code review to identify potential vulnerabilities. Following the initial manual inspection, all flagged issues were thoroughly re-examined and re-tested to confirm their validity and ensure accuracy in our final assessment.



## 1.1 Scope

### 1.1.1 In Scope

The audit comprehensively covered the Pumpkin.fun Protocol, which is designed to facilitate the creation, trading, and staking of digital assets using bonding curves, automated liquidity management, and staking rewards. The protocol consists of three main components: Bonding Curve, Liquidity Manager, and Pumpkin Staking, each playing a crucial role in its functionality. The primary areas assessed in this audit are as follows:

**Liquidity Manager Program**

The Liquidity Manager Program is responsible for automating and optimizing liquidity operations across multiple pools. It ensures smooth interactions with AMMs and liquidity providers by efficiently handling fee collection, reward distribution, and liquidity adjustments.

**Key Functions:**

- Fee Collection & Allocation: Collects trading fees from external liquidity pools like Raydium CLMM, then splits them into protocol fees, staking rewards, and creator incentives.
- Staking Reward Management: Automates the deposit and transfer of staking rewards based on predefined conditions.
- Liquidity Optimization: Ensures the protocol remains liquid by managing and redistributing funds efficiently.

**Pumpkin Staking Program**

The Pumpkin Staking Program allows users to stake tokens and earn rewards securely. It automates reward distribution, manages liquidity, and ensures fair incentives using a governance-controlled system.

**Key Functions:**

- Staking & Rewards: Users stake tokens into pools and earn rewards based on duration and amount staked. Rewards are automatically calculated and distributed.
- Governance-Controlled Updates: Only authorized governance accounts can update rewards, parameters, and addresses. The Liquidity Manager ensures updates happen efficiently when certain thresholds are met.
- Secure Liquidity Management: Utilizes SolVault to securely store and distribute SOL rewards, preventing reward resets and ensuring fair fund allocation.

**Pumpkin Bonding Curve**

The Pumpkin Bonding Curve is an automated market-making mechanism that enables users to create and trade tokens against SOL. It functions as a constant product market maker with virtual reserves, allowing for seamless token launches and decentralized trading.

**Key Functions:**

- Token Creation: Users can launch new tokens with customized supply, metadata, and bonding curve mechanics.
- Automated Market Making (AMM): All swaps between SOL and tokens follow the bonding curve pricing model, ensuring efficient price discovery.
- Trading Fees & Distribution: SOL Fees are split between creators, the protocol, and PKIN staking rewards and Token Fees are allocated to an index fund to support ecosystem growth.
- Liquidity Bootstrapping: Once all tokens are sold, a migrator bot moves liquidity from the bonding curve to an external DEX, enabling continued trading.

**Contracts in Scope:**

1. **Bonding Curve** bonding-curve/src/*
2. **Pumpkin Staking** pumpkin_staking/src/*
3. **Pumpkin Liquidity Manager** pumpkin-liquidity-manager/src/*

**Commit Hash:**

**Bonding Curve :** 66c0eef7db4a591dd451766cc097bfe2e95ee424

**Staking :** e6dc6489e0b45d1e4d4bb3ca2d6f837b90bd1891

**Liquidity Manager :** b1f2083f69c138cafae2da716cdc68e662b06c26

### 1.1.2 Out of Scope

All features or functionalities not delineated within the “In Scope” section of this document shall be deemed outside the purview of this audit.This exclusion also particularly applies to the backend operations of the Pumpkin.fun.

## 1.2 Methodology

The codebase was audited through a structured and systematic approach Over a span of 6 days. Starting with the recon phase, a basic understanding was developed, and the auditor worked on developing presumptions for the developed codebase and the relevant documentation/whitepaper. Furthermore, the audit moved on with the manual code reviews to find logical flaws in the codebase complemented with code optimizations, software, and security design patterns, code styles and best practices.

### Centralization Risks and Assumptions

The Pumpkin Protocol consists of three key interconnected components:

- **Bonding Curve Program** (Handles token creation and bonding curve trading)
- **Staking Program** (Manages staking rewards and pools)
- **Liquidity Manager Program** (Oversees fee collection and distribution)

These programs interact with each other to facilitate smooth liquidity provisioning, reward allocation, and governance. However, due to their reliance on centralized roles for initialization and configuration, certain risks and assumptions must be considered.

During this audit, we assume that the designated admin(s) and authorized signers will correctly initialize and manage the protocol by:

**1. Accurate Initialization & Updates Across All Programs**

The admins for each program—Bonding Curve's `initialize_global_config`, Staking's initialize, and Liquidity Manager's `initialize_global_state` will set valid addresses for fee recipients, governance signers, authentication roles, pool creation signers, and reward updation signers. This ensures that only authorized accounts perform sensitive actions and that subsequent operations reference correct configuration data.

**2.Bonding Curve Program: Migrator's Control Over Liquidity**

The migrator bot (configured via `initialize_global_config` and `update_global_config`) will properly handle liquidity transitions removing liquidity from the bonding curve only after token sales are complete via a predefined trheshold and then moving it to Raydium in a secure manner that prevents exploitation or fund loss.

**3. Liquidity Manager Program: Token State Initialization & Authenticator Role**

The authenticator (migrator bot) in the Liquidity Manager Program will use `initialize_token_state` responsibly, ensuring each token state is assigned to the correct creator (i.e. the same entity that launched the token on the Bonding Curve).

**4. Staking Program: Handling of Unclaimed Rewards**

A risk arises in the staking system when rewards accumulate before any users have staked. In such cases, these undistributed rewards remain within the SolVault rather than being allocated to users. To address this, a mitigation mechanism has been implemented in a off-chain component where a small dust staking transaction is included with the first reward update. This ensures that if no other stakes exist, the accumulated rewards are still assigned to an account. Once the dust reward reaches a meaningful amount, it is collected and redistributed to supplement future staking rewards.

### 1.3 Questions for Security Assessment

These are the main dynamic questions for security assessment, but the evaluation is not limited to these alone.

1. How does the contract handle staking and unstaking operations to ensure that rewards are properly allocated and no unauthorized withdrawals or miscalculations occur?
2. What access control mechanisms are implemented to protect administrative functions, such as updating reward parameters, modifying liquidity settings, and managing staking pools?
3. Does the contract properly validate and manage unclaimed rewards, ensuring that staking rewards are not lost or left in an inaccessible state?
4. How does the system prevent misconfigurations in bonding curve mechanics, ensuring fair pricing and liquidity migration when tokens graduate to external DEXs?
5. When users swap tokens for SOL (or vice versa), how are swap fees calculated, allocated, and verified especially regarding the allocation to creators, protocol, and the index fund? Additionally, how does the system ensure threshold-based events (like bonding curve completion) are accurately triggered?
6. In the process of collecting Raydium CLMM fees, distributing them among the protocol, creator, and staking pools, how does the contract verify that the correct entities receive the correct amounts, and that no double claiming or improper distribution occurs?
7. Does the program ensure that rewards are deposited into the intended staking pool for both PKIN and each specific token, preventing funds from being routed to an incorrect pool?.
8. How does the contract verify that accrued rewards meet the required thresholds before deposit, avoiding scenarios where rewards are either deposited prematurely or delayed indefinitely?
9. How does the program ensure that the 1 billion total token supply minted at creation is properly allocated to the bonding curve vault and cannot be hijacked or re-minted by an unauthorized entity?
10. In the constant product AMM model, how are real versus virtual reserves tracked to prevent manipulation or inaccurate price settings?.

### 1.4 Status Descriptions

**Acknowledged:** The issue has been recognized and is under review. It indicates that the relevant team is aware of the problem and is actively considering the next steps or solutions.

**Fixed:** The issue has been addressed and resolved. Necessary actions or corrections have been implemented to eliminate the vulnerability or problem.

**Closed:** This status signifies that the issue has been thoroughly evaluated and acknowledged by the development team. While no immediate action is being taken.

## 1.5 Summary of Findings Identified

| S.NO | SEVERITY | FINDINGS | STATUS |
|---|---|---|---|
| #1 | LOW | Lack of Two-Step Ownership Transfer in Administrative Functions | ACKNOWLEDGED |
| #2 | INFO | Missing Event Emission | ACKNOWLEDGED |
| #3 | INFO | Missing Bonding Curve PDA Closure Function | ACKNOWLEDGED |

# 2 Findings and Risk Analysis

## 2.1 Lack of Two-Step Ownership Transfer in Administrative Functions

**Severity:** Low

**Status:** Acknowledged

**Location** :

1. pumpkin_staking/src/instructions/governance/update_contract_metadata.rs
2. pumpkin-liquidity-manager/src/instructions/admin/update_global_config.rs

**Description** :

The update_global_config function allows the protocol admin to update key protocol settings, including transferring administrative control to a new admin via the new_admin field. However, this transfer is executed in a single step, meaning the current admin can directly change the admin address without requiring confirmation from the new admin.

Additionally, a similar issue exists in another instance where governance address updates occur without a two-step process. The update_contract_metadata_handler function permits updating the governance address without requiring explicit confirmation from the new governance entity. This creates a scenario where an unintended or malicious address could gain control over contract metadata updates without verification

**Recommendation** :

To mitigate this issue, implement a two-step ownership transfer mechanism:

- **Nomination Step:** The current admin sets a pending_admin field rather than replacing the admin instantly.
- **Confirmation Step:** The new admin must explicitly accept the role by invoking a separate confirmation function.

### 2.2 Missing Event Emission

**Severity:** Info

**Status:** Acknowledged

**Location** :

1. bonding-curve/src/instructions/admin/update_global_config.rs
2. pumpkin-liquidity-manager/src/instructions/admin/update_global_config.rs

**Description** :

The update_global_config function is used to update important protocol parameters in the global state account. However, it doesn't emit any events when these important changes occur. Adding event emission is a best practice for on-chain transparency and to help off-chain services track protocol configuration changes.

**Recommendation** :

Add a GlobalConfigUpdated event structure and emit it when the “**update_global_config**” is called.

### 2.3 Missing Bonding Curve PDA Closure Function

**Severity:** Info

**Status:** Acknowledged

**Location** :

pumpkin-bonding-curve/programs/bonding-curve/src/lib.rs

**Description** :

While the protocol provides mechanisms for creating the bonding curve, depositing PKIN staking rewards, and removing liquidity (allowing fee collection by the creator and the protocol), there is no function to close the BondingCurve PDA once its lifecycle is complete. As a result, even after liquidity migration and fee collection, the BondingCurve account remains on-chain and continues to hold its rent-exempt lamports indefinitely.

**Recommendation** :

Add a dedicated closure function for the BondingCurve PDA that can be called once its purpose (i.e. liquidity migration and fee collection) is complete.

1. Verify that the bonding curve is no longer active (e.g., ensure that liquidity migration has been executed and that the bonding curve is marked as complete).
2. Transfer any remaining lamports from the BondingCurve account to a designated receiver (e.g., the creator, migrator, or protocol treasury).
3. Close the BondingCurve PDA account, reclaiming the rent-exempt lamports.

**Disclaimer:**

The smart contracts provided by the client with the purpose of security review have been thoroughly analyzed in compliance with the industrial best practices till date w.r.t. Smart Contract Weakness Classification (SWC) and Cybersecurity Vulnerabilities in smart contract code, the details of which are enclosed in this report.

This report is not an endorsement or indictment of the project or team, and they do not in any way guarantee the security of the particular object in context. This report is not considered, and should not be interpreted as an influence, on the potential economics of the token (if any), its sale, or any other aspect of the project that contributes to the protocol's public marketing.

Crypto assets/ tokens are the results of the emerging blockchain technology in the domain of decentralized finance and they carry with them high levels of technical risk and uncertainty. No report provides any warranty or representation to any third-party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the reports in any way, including to make any decisions to buy or sell any token, product, service, or asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and is not a guarantee as to the absolute security of the project.

Smart contracts are deployed and executed on a blockchain. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. The scope of our review is limited to a review of the programmable code and only the programmable code, we note, as being within the scope of our review within this report. The smart contract programming language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer or any other areas beyond the programming language's compiler scope that could present security risks.

This security review cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While BlockApex has done their best in conducting the analysis and producing this report, it is important to note that one should not rely on this report only - we recommend proceeding with several independent code security reviews and a public bug bounty program to ensure the security of smart contracts