AstraSec

Lynx Protocol Security Audit Report

March 14, 2025

Contents

1 Introduction

1.1 About Lynx Protocol

1.2 Source Code

2 Overall Assessment

3 Vulnerability Summary

3.1 Overview

3.2 Security Level Reference

3.3 Vulnerability Details

4 Appendix

4.1 About AstraSec

4.2 Disclaimer

4.3 Contact

1 Introduction

1.1 About Lynx Protocol

Lynx Protocol is a gasless Perpetual Futures DEX that provides traders with exposure to a diverse range of assets, including cryptocurrencies, forex, and commodities, using any token as collateral. Its innovative model mitigates price volatility risk for collateral assets, allowing governance tokens, memecoins, and stablecoins to be used as collateral without introducing additional liquidation risks.

This audit focuses exclusively on the Solana and token sweeping functionality within the Lynx Protocol smart contracts.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/lynx-protocol/lynx-core-solana.git
- CommitID: 524056a

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/lynx-protocol/lynx-core-solana.git
- CommitID: dac8a4f

Note this audit only covers the sweep_native.rs and sweep_tokens.rs contracts.

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the Lynx protocol. Throughout this audit, we identified a total of 2 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	–	–	–	–
High	–	–	–	–
Medium	–	–	–	–
Low	–	–	–	–
Informational	2	–	–	2
Undetermined	–	–	–	–

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~I-1~~ Enchanced Sanity Checks in SweepNative::apply()

~~I-2~~ Redundant State/Code Removal

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Acknowledged
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and re-mediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

3.3.1 [I-1] Enhanced Sanity Checks in SweepNative::apply()

Target	Category	IMPACT	LIKELIHOOD	STATUS
sweep_native.rs	Coding Practices	NA	NA	Addressed

The SweepNative::apply() function lacks thorough validation checks, particularly concerning rent exemption requirements. The code directly modifies the lamports of the from_info and dest accounts without ensuring that the remaining lamports in the from_info account meet the minimum rent-exempt threshold. This oversight could result in the from_info account being deallocated if its lamports balance falls below the rent-exempt requirement after the transfer.

lynx-core-solana-main - sweep_native.rs

```rust
pub fn apply(ctx: &mut Context<SweepNative>, params: &SweepNativeParams) -> Result<()> {
    let from_info = ctx.accounts.oft_store.to_account_info();
    require!(
        from_info.lamports() >= params.amount,
        OFTError::InsufficientBalance
    );
    **from_info
        .try_borrow_mut_lamports()? -= params.amount;
    **ctx
        .accounts
        .dest
        .to_account_info()
        .try_borrow_mut_lamports()? += params.amount;
    Ok(())
}
```

Remediation Add sanity checks to ensure the source account's remaining lamports meet the minimum rent-exempt threshold after the transfer.

3.3.2 [I-2] Redundant State/Code Removal

Target	Category	IMPACT	LIKELIHOOD	STATUS
sweep_tokens.rs	Coding Practices	NA	NA	Addressed

While examining the current implementation, we notice the token_escrow account within the SweepTokensRenounceInit struct appears to be redundant. This account is defined and constrained but is not utilized anywhere in the context of the logic. Retaining this unused account in the struct introduces unnecessary complexity, elevates computational overhead during validation, and may create confusion for developers maintaining the codebase. Furthermore, the token_escrow account within the SweepTokensRenounce struct is similarly redundant.

lynx-core-solana-main - sweep_tokens.rs

```
85 pub struct SweepTokensRenounceInit<'info> {
86     ...
87     #[account(
88         address = oft_store.token_escrow,
89         token::authority = oft_store,
90     )]
91     pub token_escrow: InterfaceAccount<'info, TokenAccount>,
92     ...
93 }
```

Remediation Remove the redundant token_escrow account from the SweepTokens-RenounceInit and SweepTokensRenounce structs to reduce complexity and improve code clarity.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI