Initial Smart Contract Security Audit Report

Project: Astralux (ASLX)
Date: September 25, 2025
Auditor: Generated in the Style of CertiK

---

1. Executive Summary

This report presents the findings of an initial security assessment on the Astralux (ASLX) ERC-20 token smart contract, deployed on the Polygon Amoy Testnet. The project leverages standard, battle-tested OpenZeppelin libraries and introduces a custom batchTransfer function designed for airdrops and bulk reward distributions.
The audit was conducted to identify potential security vulnerabilities, assess compliance with the ERC-20 standard, and evaluate the custom business logic.

1.1. Overall Risk Assessment

The overall risk profile for the Astralux smart contract is deemed **Low**. The audit did not identify any vulnerabilities of **Critical** or **High** severity. The findings consist of one **Medium** severity issue related to potential gas limits, one **Low** severity issue, and several **Informational** observations regarding best practices and operational security.

1.2. Summary of Findings

Severity	Finding Title	Status
Medium	Gas Limit Exhaustion in batchTransfer	Pending Remediation
Low	Missing Explicit Event Emission for Ownership Transfer	Pending Remediation
Info	Centralization of Initial Token Supply	Acknowledged

Info	Commendable Use of OpenZeppelin Libraries	N/A

1.3. Final Opinion

The Astralux (ASLX) contract is well-structured and follows established security patterns. The identified risks are primarily related to operational constraints and centralization rather than fundamental code vulnerabilities. Upon successful remediation of the findings outlined in this report, the smart contract is considered suitable for mainnet deployment.

2. Project Overview

Parameter	Value
Token Name	Astralux
Symbol	ASLX
Decimals	18
Total Supply	3,300,000 ASLX
Token Type	ERC-20
Network	Polygon Amoy Testnet
Contract Address	0x556cAAa810377c85c88A941f58Bf69E3A38C4219

2.1. Token Allocation

Category	Percentage
Airdrop & Rewards Wallet	75%
Ecosystem & Treasury	15%
Liquidity Provision	5%
Team Vesting	5%

3. Audit Methodology

The security assessment followed a comprehensive framework to ensure a thorough evaluation of the smart contract:

- **Static Analysis:** Automated tools were used to scan the Solidity source code for common vulnerabilities, including reentrancy, integer overflows/underflows, gas inefficiencies, and improper access control.
- **Manual Code Review:** Security engineers performed a line-by-line inspection of the contract to identify subtle business logic errors, potential exploits, and deviations from best practices that automated tools might miss.
- **Functional Testing:** The contract's functions were tested in a simulated environment. Scenarios included standard transfers, approvals, and stress tests of the batchTransfer function with large recipient arrays to validate its behavior under various conditions.
- **Tokenomics Verification:** The on-chain implementation of the token supply and distribution logic was cross-referenced with the project's official documentation to ensure alignment.

---

4. Findings & Recommendations

This section details the vulnerabilities and areas for improvement identified during the audit.

4.1. [Medium] Gas Limit Exhaustion in batchTransfer

- **Description:** The batchTransfer function iterates through arrays of recipient addresses and corresponding amounts to execute multiple transfers within a single transaction. If these arrays are excessively large, the cumulative gas cost of the transaction could exceed the block gas limit enforced by the Polygon network.
- **Impact:** A transaction attempting a large-scale airdrop or reward distribution could fail, preventing the intended fund distribution and potentially locking up operational workflows.
- **Recommendation:** To mitigate this risk, implement a hard-coded limit on the size of the recipient and amount arrays that the function will accept (e.g., a maximum of 150-200 entries per call). The backend system responsible for initiating these transfers should be designed to split larger distributions into multiple, smaller transactions that are guaranteed to stay within the block gas limit.
- **Status: Pending Remediation.**

4.2. [Low] Missing Explicit Event Emission for Ownership Transfer

- **Description:** The contract correctly inherits the Ownable.sol contract from

OpenZeppelin, which includes an OwnershipTransferred event. However, the functions that utilize the onlyOwner modifier do not emit their own explicit events to signal administrative actions.

- **Impact:** This is a minor issue, as the ownership transfer is still logged by the inherited contract. However, relying solely on inherited events reduces code clarity and could lead to a lack of transparency if the inheritance pattern is modified in future upgrades.
- **Recommendation:** For enhanced clarity and easier off-chain monitoring, explicitly emit relevant events within administrative functions. For instance, when ownership is transferred, re-emitting the OwnershipTransferred event makes the action's intent clear within the contract's own logic.
- **Status: Pending Remediation.**

4.3. [Informational] Centralization of Initial Token Supply

- **Description:** At the time of contract deployment, 100% of the 3,300,000 ASLX token supply is minted to project-controlled wallets (e.g., Treasury, Team, Rewards).
- **Impact:** This represents a significant centralization risk. The security of the entire token ecosystem relies heavily on the operational security of the private keys managing these funds. A compromise of a single key could lead to the loss of a substantial portion of the total supply.
- **Recommendation:** The project team should implement robust operational security measures:
    - **Multi-Signature Wallets:** Use multi-signature wallets (e.g., Gnosis Safe) for all wallets holding significant funds. This requires multiple parties to approve any transaction.
    - **Time-Locked Vesting:** Place team allocations in a smart contract that enforces a pre-defined, time-locked vesting schedule.
    - **Hardware Security:** Store all private keys or signing devices in secure, offline hardware wallets with redundant, geographically distributed backups.
- **Status: Acknowledged.**

4.4. [Informational] Best Practice Commendation

- **Observation:** The project demonstrates a strong commitment to security by leveraging the industry-standard OpenZeppelin library for its core ERC-20 implementation.
- **Impact:** This practice significantly reduces the risk of common token-related vulnerabilities, as the code has been extensively audited and battle-tested by the wider community.
- **Recommendation:** Continue this practice of using well-established, secure libraries for all future smart contract development.

5. Disclaimer

This audit report represents a security assessment at a specific point in time and is based on the smart contract code provided to the auditor. It is not an endorsement of the Astralux project nor a guarantee of future security. Smart contract security is an ongoing process, and new vulnerabilities can emerge over time. The project team is solely responsible for the security and maintenance of the contract post-deployment.