



BACKEND SECURITY AUDIT

Luxxcoin

Backend

CLIENT

Luxxcoin

PLATFORM

Backend service

REVIEW

November 2025

DOCUMENT

Document information

Project	Luxxcoin, Backend
Client	Luxxcoin
Platform	Backend service
Language	Not recorded in the original report
Repository	Luxxcoin-team/Luxx-Backend: src
Commit	Not recorded in the original report
Methods	Comprehensive manual code review, static analysis, threat-modeling, and best-practice compliance verification, following OWASP ASVS and OWASP Risk Rating Methodology
Timeframe	November 2025
Status at issue	Under review

Document history

VERSION	DATE	CHANGE
1.0	November 2025	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	3
02	Scope	4
03	Methodology	5
04	Severity and status	6
05	Findings summary	7
06	Detailed findings	8
	C-01 Authentication Bypass via User-Controlled Public Key in Ed25519Auth	8
	Disclaimer	10

Executive summary

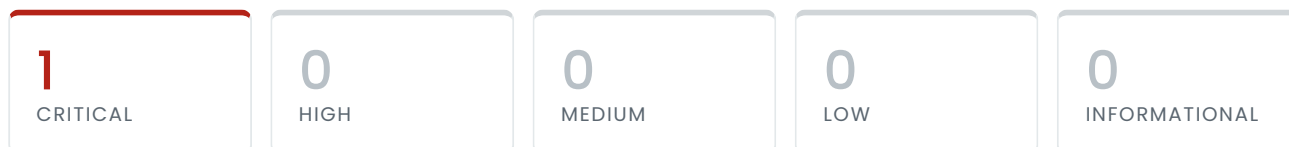
About Luxxcoin

Luxxcoin (\$LUX) is a Solana-based MemeFi project that turns meme culture into a rules-based, on-chain competition. Every two months, the top 50 wallets are rewarded from a pool of 1.0 Billion \$LUX. The backend runs round processing, scoring, scheduling and payouts behind admin endpoints.

Summary of the review

Safe Edges reviewed the Luxxcoin backend (the src directory of the Luxx-Backend repository) in November 2025. One Critical issue was identified: the admin authentication middleware verified signatures against a public key supplied by the caller instead of the configured governance key, so anyone could sign their own requests and reach admin endpoints such as round processing, scoring configuration and payouts. The report was issued while remediation was under review, so no resolution status was recorded.

Findings by severity



Findings by status

STATUS	ISSUES
● Not recorded	1

Scope

Repository	Luxxcoin-team/Luxx-Backend: src
Commit	Not recorded in the original report
Platform	Backend service
Language	Not recorded in the original report

IN-SCOPE FILES

```
src
```

Only the code at the commit above was reviewed. Changes made after this commit are not covered by this report.

Methodology

Methods recorded for this engagement: Comprehensive manual code review, static analysis, threat-modeling, and best-practice compliance verification, following OWASP ASVS and OWASP Risk Rating Methodology.

Manual review

Auditors read the in-scope code line by line, tracing control flow, state changes and trust boundaries against the intended behaviour described by the client.

Threat modelling

We map privileged roles, external dependencies and value flows, then test the assumptions the code makes about each of them.

Functional and automated testing

Where tooling exists for the language and chain, we run static analysis and targeted tests to confirm suspected issues and check edge cases.

Fix review

After the client addresses the findings, we review the updated code and record the final status of each issue.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
C-01	Authentication Bypass via User-Controlled Public Key in Ed25519Auth	Critical	● Not recorded

Detailed findings

C-01 Authentication Bypass via User-Controlled Public Key in Ed25519Auth

Severity **Critical** Status ● Not recorded Location Ed25519Auth middleware

CLASSIFICATION

- Impact: Access control compromise, admin endpoints fully exposed
- Category: Authentication bypass / Broken cryptographic design
- OWASP: A07 – Identification & Authentication Failures

SUMMARY

The ed25519Auth middleware accepts the following headers:

- X-Gov-Public-Key
- X-Gov-Signature
- X-Gov-Challenge
- X-Gov-Timestamp

It verifies that the provided signature is valid for the message challenge:timestamp using the public key supplied by the caller. The configured trust anchor (`config.auth.govPublicKey`) is never used. As a result, an attacker can:

1. Generate any Ed25519 key pair.
2. Choose an arbitrary challenge string and a current timestamp.
3. Sign challenge:timestamp with their private key.
4. Call any admin route, providing the attacker's public key in X-Gov-Public-Key.

Verification succeeds because the signature is valid for the attacker-supplied key – even though that key is not trusted by the server. This yields a complete authentication bypass.

EXAMPLE (EXCERPT OF VULNERABLE LOGIC)

```
ed25519Auth(req, res, next) {  
  const signature = req.headers['x-gov-signature'];  
  const publicKey = req.headers['x-gov-public-key'];  
  const challenge = req.headers['x-gov-challenge'];  
  const timestamp = req.headers['x-gov-timestamp'];  
  // ...  
  const message = `${challenge}:${timestamp}`;  
  const isValid = this.verifyEd25519Signature(message, signature, publicKey);  
  // ...  
}
```

IMPACT

All admin-only endpoints (e.g. POST /rounds/:id/process, scoring/config endpoints, scheduler triggers, payouts) become accessible to anyone who can send forgable headers. Attackers can finalize rounds, change scoring weights, trigger payouts, or perform other privileged actions without authorization. This represents a full loss of access control for governance/admin functionality.

ROOT CAUSE

The middleware verifies signatures using client-supplied public keys instead of the server-configured trusted authority key. This removes the trust anchor that makes digital signatures meaningful: signatures are only authentic when checked against a public key the server already trusts.

RECOMMENDED FIXES

1. Pin the trusted public key never accept the public key from the client. Always verify signatures against

```
if (publicKey !== this.govPublicKey) {
  return res.status(403).json({
    success: false,
    error: 'Invalid GOV public key',
    timestamp: new Date().toISOString(),
  });
}
```

Disclaimer

This report is based on the code at the commit listed in the Scope section and on information the client provided during the review. It reflects Safe Edges' assessment at the time of the review and is not a guarantee that the code is free of vulnerabilities.

Changes made after the reviewed commit, deployment configuration, off-chain components and third-party dependencies are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS