hexens x ZKsync

Security Review Report for ZKsync

August 2025

Table of Contents

1. About Hexens

Hexens is a pioneering cybersecurity firm dedicated to establishing robust security standards for Web3 infrastructure, driving secure mass adoption through innovative protection technology and frameworks. As an industry elite experts in blockchain security, we deliver comprehensive audit solutions across specialized domains, including infrastructure security, Zero Knowledge Proof, novel cryptography, DeFi protocols, and NFTs.

Our methodology combines industry-standard security practices combined with unique methodology of two teams per audit, continuously advancing the field of Web3 security. This innovative approach has earned us recognition from industry leaders.

Since our founding in 2021, we have built an exceptional portfolio of enterprise clients, including major blockchain ecosystems and Web3 platforms.

2. Security Review Details

▪ Review Led by

Hayk Andriasyan, Lead Security Researcher

▪ Scope

The analyzed resources are located on:

🔗 https://github.com/matter-labs/zksync-airbender/tree/audit_branch

📌 Commit: `a10cb1d4e38652eac4fcb4593bfc261d26527a98`

The issues described in this report were Acknowledged.

▪ Changelog

12 August 2025	Audit start
1 October 2025	Initial report
10 October 2025	Revision received
14 October 2025	Final report

3. Severity Structure

The vulnerability severity is calculated based on two components:

1. Impact of the vulnerability
2. Probability of the vulnerability

Impact	Probability			
	Rare	Unlikely	Likely	Very likely
Low	Low	Low	Medium	Medium
Medium	Low	Medium	Medium	High
High	Medium	Medium	High	Critical
Critical	Medium	High	Critical	Critical

- **Severity Characteristics**

Smart contract vulnerabilities can range in severity and impact, and it's important to understand their level of severity in order to prioritize their resolution. Here are the different types of severity levels of smart contract vulnerabilities:

Critical

Vulnerabilities that are highly likely to be exploited and can lead to catastrophic outcomes, such as total loss of protocol funds, unauthorized governance control, or permanent disruption of contract functionality.

High

Vulnerabilities that are likely to be exploited and can cause significant financial losses or severe operational disruptions, such as partial fund theft or temporary asset freezing.

Medium	Vulnerabilities that may be exploited under specific conditions and result in moderate harm, such as operational disruptions or limited financial impact without direct profit to the attacker.
Low	Vulnerabilities with low exploitation likelihood or minimal impact, affecting usability or efficiency but posing no significant security risk.
Informational	Issues that do not pose an immediate security risk but are relevant to best practices, code quality, or potential optimizations.

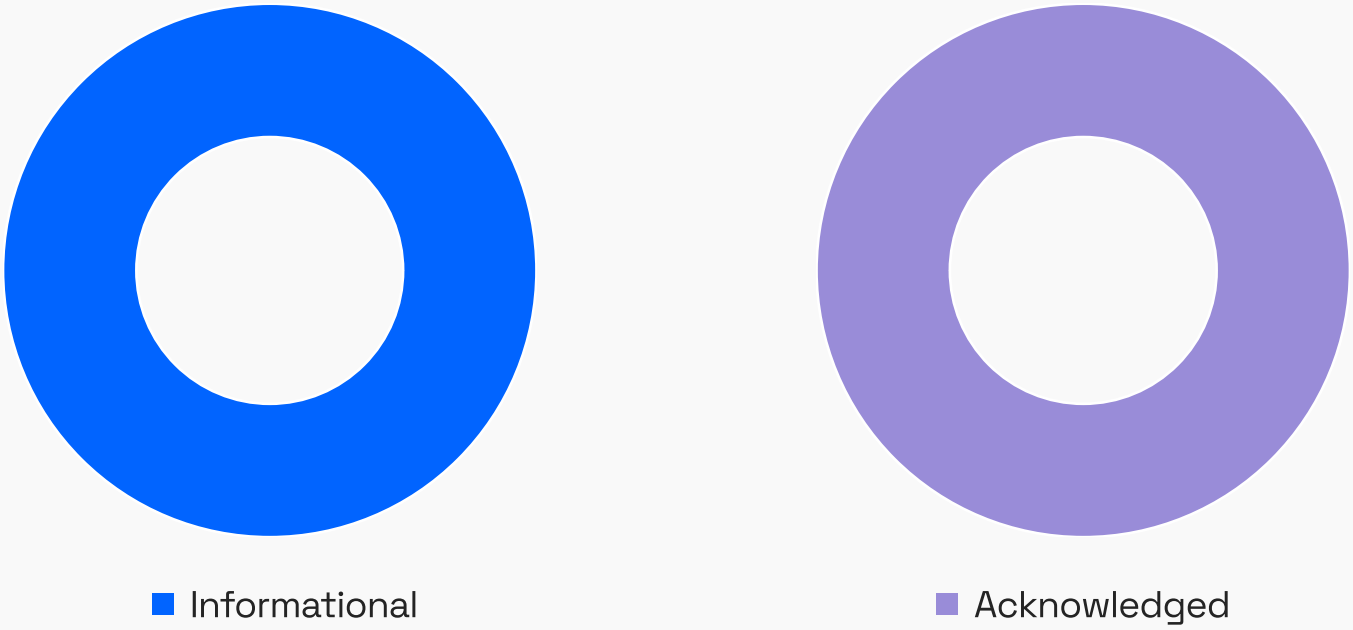
- **Issue Symbolic Codes**

Each identified and validated issue is assigned a unique symbolic code during the security research stage.

Due to the structure of the vulnerability reporting flow, some rejected issues may be missing.

4. Findings Summary

Severity	Number of findings
Critical	0
High	0
Medium	0
Low	0
Informational	1
Total:	**1**



5. Weaknesses

This section contains the list of discovered weaknesses.

ZKSVM-1 | Redundant assertion in add_constraint function

Acknowledged

Severity:	Probability:	Impact:
Informational	Likely	Informational

Path:

cs/src/cs/cs_reference.rs:#L145

Description:

The **add_constraint** function inserts a constraint object into the constraint storage, asserting that the constraint's degree is exactly two.

```
fn add_constraint(&mut self, mut constraint: Constraint<F>) {
    assert!(constraint.degree() == 2, "use `add_constraint_allow_explicit_linear`
if you need to make a variable arising from linear constraint");
    assert!(constraint.degree() <= 2);
    constraint.normalize();
    self.try_check_constraint(&constraint);
    self.constraint_storage.push((constraint, false));
}
```

The second assertion is redundant, as the first assertion already guarantees that the degree is equal to 2.

Remediation:

Remove redundant assertion.


hexens
x
ZKsync