Guvenkaya® The Bedrock of Security

Sailor Lend

Sailor Lend Web Application Security Review

Lead Security Engineer: Michal Bajor
Date of Engagement: 10th December 2025 - 29th December 2025
Visit: www.guvenkaya.co

Contents

About Us

Guvenkaya is a security research firm specializing in Rust security, Web3 security of Non-EVM protocols, and Web2 security. With our expertise, we provide both security auditing services and custom security solutions

About Sailor Lend

Sailor Lend is a trustless, self-custodial DeFi lending protocol built on the NEAR blockchain. The platform enables users to obtain instant loans using BTC, ZEC, or SOL as collateral without requiring KYC verification. Users can deposit stablecoins for lending to earn interest, while borrowers can leverage their crypto assets to access liquidity. The protocol implements liquidation mechanisms, interest rate calculations, and Pyth price oracle integrations.

Audit Results

Guvenkaya conducted a comprehensive security assessment of the Sailor Lend web application. During this engagement, 2 findings were reported: 1 Critical and 1 High severity issue. The critical finding includes a remote code execution vulnerability through React2Shell (CVE-2025-55182) affecting React Server Components. All findings have been addressed by the Sailor Lend team.

Project Scope

Sailor Lend Web Application

Files	Link
Sailor Lend Web Application	https://github.com/Satoshi-Port/satoshi-port/tree/93fbf4f33c97999cb74ff109f355b6c73bc722f8/src

Out of Scope

The audit included reviewing the code for security vulnerabilities, coding practices, and architecture. The audit does not include a review of the dependencies or external contracts such as Pyth Oracle and FT contracts.

Timeline

Start of the audit	Draft report	Final report
10th December 2025	29th December 2025	17th March 2026

Methodology

RESEARCH INTO PROJECT ARCHITECTURE

PREPARING ATTACK VECTORS

SETTING UP AN ENVIRONMENT

MANUAL CODE REVIEW OF THE CODE

ASSESSMENT OF WEB APPLICATION SECURITY ISSUES

ASSESSMENT OF ARITHMETIC ISSUES

BUSINESS LOGIC VULNERABILITY ASSESSMENT

Severity Breakdown

01. Likelihood Ratings

Likely: The vulnerability is easily discoverable and not overly complex to exploit.
Possible: The vulnerability presents some challenges either in discovery or in the complexity of the attack.
Rare: The vulnerability is either very difcult to discover or complex to exploit, or both.
This matrix provides a nuanced view, taking into account both the ease of discovering a vulnerability and the complexity involved in exploiting it.

02. Impact

Severe: Exploitation could result in critical loss or compromise, such as full system control, substantial financial loss, or severe reputational damage.
Moderate: Exploitation may lead to limited data loss, partial compromise, moderate financial impact, or noticeable degradation of services.
Negligible: Exploitation has minimal impact, such as minor data exposure without significant consequences or slight inconvenience without substantial disruption.

03. Severity Ratings

Critical: Assigned to vulnerabilities with severe impact and a likely likelihood of exploitation.
High: For vulnerabilities with either severe impact but only a possible likelihood, or moderate impact with a likely likelihood.
Medium: Used for vulnerabilities with severe impact but a rare likelihood, moderate impact with a possible likelihood, or negligible impact with a likely likelihood.
Low: For vulnerabilities with moderate impact and rare likelihood, or negligible impact with a possible likelihood.
Informational: The lowest severity rating, typically for vulnerabilities with negligible impact and a rare likelihood of exploitation.

CRITICAL **HIGH** **MEDIUM** Low Informational

Likelihood Matrix:

Attack Complexity \ Discovery Ease	Obvious	Concealed	Hidden
Complex	Possible	Rare	Rare
Moderate	Likely	Possible	Rare
Straightforward	Likely	Possible	Possible

Likelihood/Impact Matrix:

Likelihood \ Impact	Severe	Moderate	Negligible
Likely	**CRITICAL**	**HIGH**	**MEDIUM**
Possible	**HIGH**	**MEDIUM**	Low
Rare	**MEDIUM**	Low	Informational

Findings Summary

01. Remediation Complexity: This measures how difcult it is to fx the vulnerability once it has been identifed.

Simple: Patches or fixes are readily available and easily implemented.

Moderate: Requires some time and resources to remediate, but well within the capabilities of most organizations.

Difficult: Remediation requires significant resources, specialized skills, or substantial changes to systems or architecture.

02. Status: This measures how difcult it is to fx the vulnerability once it has been identifed.

Not Fixed: Indicates that the vulnerability has been identifed but no remedial action has been taken yet. This status is crucial for newly discovered vulnerabilities or those awaiting prioritization.

Fixed: This status is applied when the vulnerability has been successfully remediated. It implies that appropriate measures (like patching, confguration changes, or architectural modifcations) have been implemented to resolve the issue.

Acknowledged: This status is used for vulnerabilities that have been recognized, but for various reasons (such as risk acceptance, cost, or other business decisions), have not been fxed. It indicates that the risk posed by the vulnerability is known and has been consciously accepted.

Scheduled: This status indicates that the vulnerability has been acknowledged and a plan is in place to fix it in the future. It signifies that while remediation hasn't yet occurred, the issue has been prioritized and is part of the planned development roadmap.

Finding	Impact	Likelihood	Severity	Remediation Complexity	Remediation Status
GUV-1: Vulnerable to React2Shell	Severe	Likely	CRITICAL	Simple	Fixed
GUV-2: Hardcoded Password in Source Code	Moderate	Likely	HIGH	Simple	Fixed

Findings Details

GUV-1 Vulnerable to React2Shell - Critical

The application is vulnerable to React2Shell (CVE-2025-55182), a critical remote code execution (RCE) vulnerability affecting React Server Components. The project currently uses React 19.0.0, which falls within the vulnerable version range (19.0.0 through 19.2.0).

This vulnerability stems from unsafe deserialization within React's "Flight" protocol, allowing unauthenticated attackers to execute arbitrary code on the server by sending specially crafted HTTP requests. The application uses Next.js 15.1.4 with the App Router, which relies on React Server Components and is therefore affected.

The vulnerability was confirmed using https://github.com/assetnote/react2shell-scanner tool.

Recommendation

We recommend updating the react version to a non-vulnerable version.

Remediation - Fixed

Sailor Lend team has fixed this issue in commit
3446235a0abc82377a266bc8c8a056c3523130fc

GUV-2 Hardcoded Password in Source Code - High

The source code contains a hardcoded password that is checked before accessing the web application.

Password Check

```
const passwordIsCorrect = (value: string) => {
  return value === "satoshi";
};
```

Recommendation

We recommend avoiding hard-coding sensitive values in the code. The best approach would be to send the password to the backend and do the hash-based comparison there.

Remediation - Fixed

Sailor Lend team has fixed this issue in commit
3446235a0abc82377a266bc8c8a056c3523130fc