Zellic

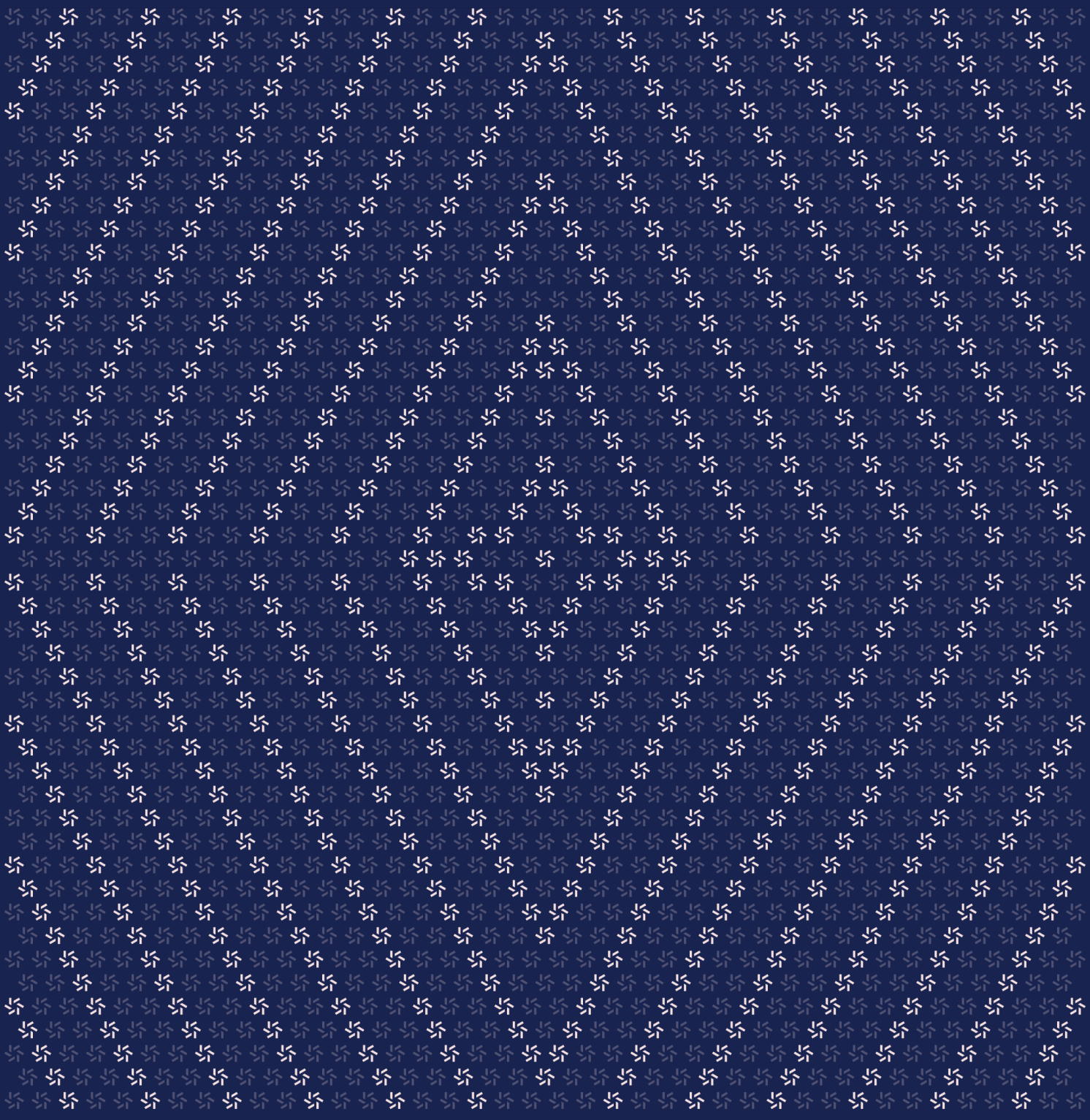
Prepared for
Arun Devabhaktuni
SatLayer

Prepared by
Junghoon Cho
Junyi Wang
Zellic

July 24, 2024

SatLayer Pool

Smart Contract Security Assessment



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for SatLayer from July 23rd to July 24th, 2024. During this engagement, Zellic reviewed SatLayer Pool's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Are there issues in the protocol math or logic that lead to loss of funds?
- Does SatLayer Pool have any centralization risks?
- Is SatLayer Pool secure against common smart contract vulnerabilities?

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Migrator contract
- Front-end components
- Infrastructure relating to the project
- Key custody

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. Results

During our assessment on the scoped SatLayer Pool contracts, we discovered one finding, which was informational in nature.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	1

2. Introduction

2.1. About SatLayer Pool

SatLayer contributed the following description of SatLayer Pool:

> SatLayer is a universal security layer building upon Babylon to enable Bitcoin to be restaked to secure AVS applications.

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood.

We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

2.3. Scope

The engagement involved a review of the following targets:

SatLayer Pool Contracts

Type	Solidity
Platform	EVM-compatible
Target	deposit-contract-scoping
Repository	https://github.com/satlayer/deposit-contract-scoping ↗
Version	`747039458a407b3989a9b715f68a741365369ef8`
Programs	`ReceiptToken.sol` `SatlayerPool.sol`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of two person-days. The assessment was conducted by two consultants over the course of two calendar days.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald
Engagement Manager
chad@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Junghoon Cho
Engineer
junghoon@zellic.io ↗

Junyi Wang
Engineer
junyi@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

July 23, 2024	Kick-off call
July 23, 2024	Start of primary review period
July 24, 2024	End of primary review period

3. Detailed Findings

3.1. Centralization risk

Target	SatlayerPool.sol		
Category	Business Logic	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

The contract grants many permissions to the owner. These include the following:

- Pausing/unpausing the contract, potentially stopping the protocol
- Setting the migrator, which can be used to steal funds from unsuspecting users
- Recovering tokens sent to the contract that are not supported — a malicious owner may not return the funds

Impact

As mentioned above, a compromised owner account can permanently halt the protocol, causing users' deposited tokens to be frozen, or maliciously configure the migrator contract to steal users' funds. This poses some centralization risk and requires extra trust from the user.

Recommendations

Protect the owner account appropriately, one suggestion being via a multisig wallet, or design the protocol to be less centralized.

Remediation

SatLayer acknowledged this risk and plans to transfer ownership of the contract to a multi-signature wallet and consider implementing a time lock governor.

4. Threat Model

This provides a full threat model description for various functions. As time permitted, we analyzed each function in the contracts and created a written threat model for some critical functions. A threat model documents a given function's externally controllable inputs and how an attacker could leverage each input to cause harm.

Not all functions in the audit scope may have been modeled. The absence of a threat model in this section does not necessarily suggest that a function is safe.

4.1. Depositing

- ☑ It must assert that the amount to be deposited is not zero.
- ☑ It must assert that deposits are not made to the zero address.
- ☑ The token to be deposited must be allowed to be staked.
- ☑ If `capsEnabled` is true, it must assert that the total staked amount of the token after a deposit does not exceed the token's cap.

4.2. Withdrawing

- ☑ It must assert that the amount to be withdrawn is not zero.
- ☑ It must assert that the user withdrawing has staked a sufficient amount of tokens.

4.3. Migrating

- ☑ The `migrator` contract address must not be set to the zero address.
- ☑ The array of tokens must not be empty.
- ☑ The array of tokens must not include any token that the user migrating has not staked.
- ☑ The array of tokens must not include duplicate tokens.

4.4. Setting migrator

- ☑ It must assert that the migrator being set is not the zero address.

4.5. Adding tokens

- ☑ It must assert that the token being added must not have been added before.

4.6. Setting staking parameters of a token

- ☑ It must assert that the token for which the parameter is being set is not the zero address.
- ☑ It must assert that the token is not already configured with the same staking state.

4.7. Recovering ERC-20

- ☑ It must assert that the token being recovered is not the zero address.

5. Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped SatLayer Pool contracts, we discovered one finding, which was informational in nature.

5.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.