Zellic

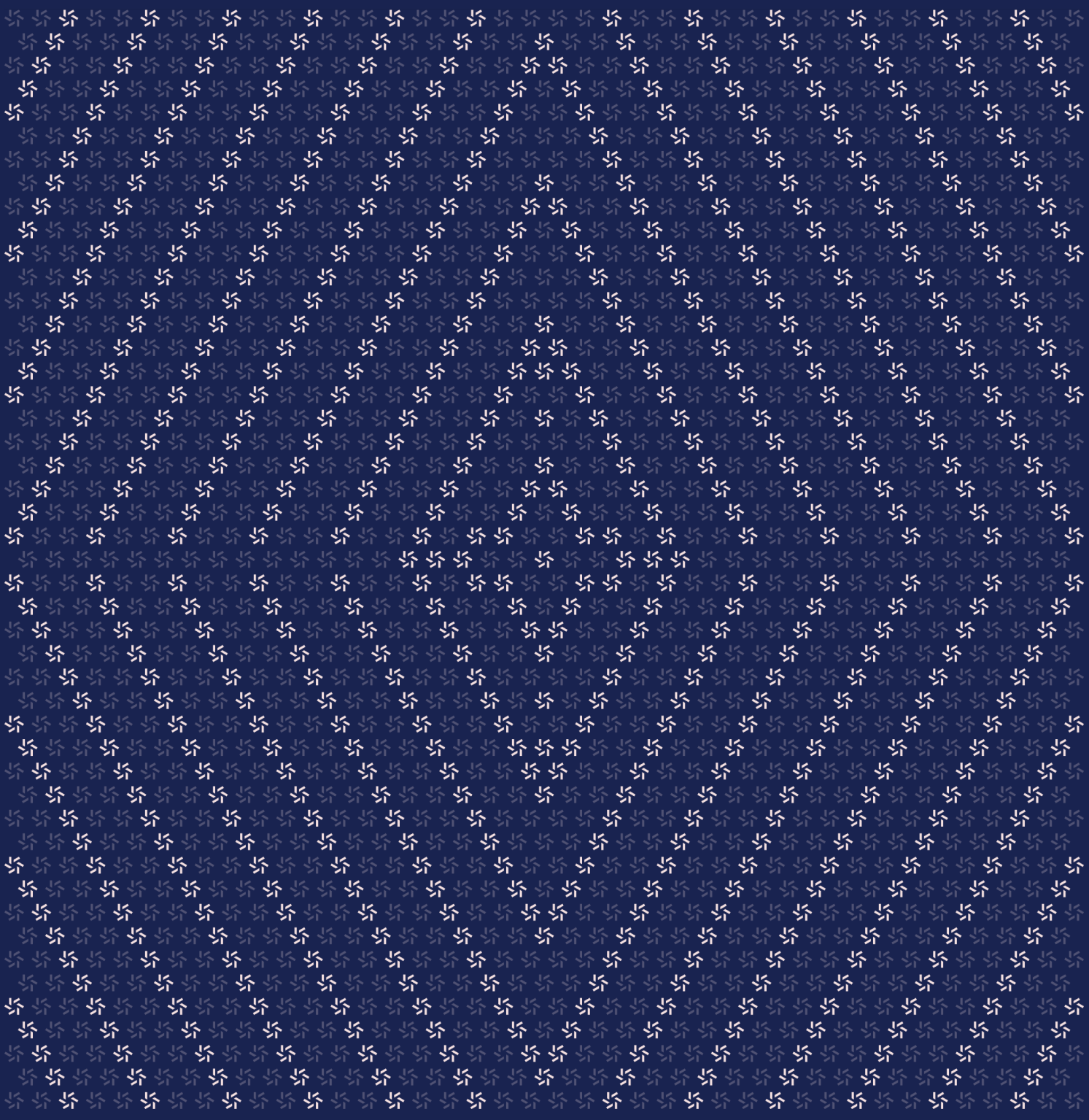
Prepared for
Franklyn Wang
Scott Becker
Liquid

Prepared by
Michael Guarino
Moritz Schneider
Zellic

August 28, 2025

LiquidMax

Web Application Security Assessment



Contents

About Zellic **3**

1. Overview **3**

1.1. Executive Summary 4

1.2. Goals of the Assessment 4

1.3. Non-goals and Limitations 4

1.4. Results 4

2. Introduction **5**

2.1. About LiquidMax 6

2.2. Methodology 6

2.3. Scope 8

2.4. Project Overview 9

2.5. Project Timeline 10

3. Detailed Findings **10**

4. Assessment Results **11**

4.1. Disclaimer 12

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 Liquid from July 28th to August 27th, 2025. During this engagement, Zellic reviewed LiquidMax'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:

- Is the referral payout system secure?
- Is the key route secure?
- Does the Swift application securely handle secrets?
- Does the Swift application contain any front-end issues that can expose customer data?
- Does the backend server securely handle authentication and authorization?
- Does the backend server contain any injection or logical vulnerabilities?
- Could an attacker abuse the Privy recovery site to recover an account that they do not own?
- Could an attacker abuse the Privy recovery site to leak information about another user?

1.3. Non-goals and Limitations

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

- Leaderboard backend
- Access codes
- 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.

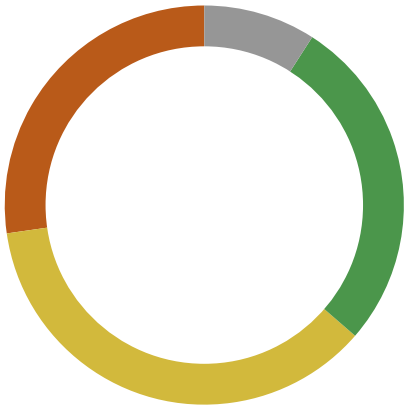
1.4. Results

During our assessment on the scoped LiquidMax contracts, we discovered 11 findings. No critical issues were found. Three findings were of high impact, four were of medium impact, three were of

low impact, and the remaining finding was informational in nature.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	3
Medium	4
Low	3
Informational	1



2. Introduction

2.1. About LiquidMax

Liquid contributed the following description of LiquidMax:

> Liquid is a mobile interface for trading spot tokens, perpetual futures and for using vaults.

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:

LiquidMax Contracts

Types	Swift, Python, TypeScript
Platform	web
Target	hyperliquid-swift-app
Repository	https://github.com/LiquidMax-dev/hyperliquid-swift-app ↗
Version	`ab673f324ee40deff9961bb18609047b2acb6748`
Programs	`Hyperliquid/**/*.swift (and associated configuration files)`
Target	liquidmax_server
Repository	https://github.com/LiquidMax-dev/liquidmax_server ↗
Version	`552b31a0507ba1baf2ef38b33f84eeababdf7406`

Programs	`**.py (excluding liquidserver/routes/leaderboard.py)`
	`**.ts`
	`prisma/**`

Target	privy-recovery-site
Repository	https://github.com/LiquidMax-dev/privy-recovery-site ↗
Version	`788289af0782a7973fcd43c629639ea3a035d413`
Programs	`src/**`

2.4. Project Overview

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

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io ↗

Chad McDonald
Engagement Manager
chad@zellic.io ↗

Pedro Moura
Engagement Manager
pedro@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Michael Guarino
Engineer
michael@zellic.io ↗

Moritz Schneider
Engineer
moritz@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

July 28, 2025	Kick-off call
July 28, 2025	Start of primary review period
August 27, 2025	End of primary review period

3. Detailed Findings

The detailed findings of this assessment have been redacted as the assessed code is not publicly available

4. Assessment Results

During our assessment on the scoped LiquidMax contracts, we discovered 11 findings. No critical issues were found. Three findings were of high impact, four were of medium impact, three were of low impact, and the remaining finding was informational in nature. At the time of our assessment, portions of the LiquidMax codebase appeared somewhat disorganized, contributing to a less-than-optimal security posture. Particularly, in the backend server repository, although code was clear, concise, and mostly well-written, there were several instances in which bugs existed and instances of code that were redundant or incorrectly commented. Although many of the aforementioned occurrences are not critical issues (or even security issues in every instance), their existence gives a combined impression that additional care should be given during the development process to ensure that any new code is adequately reviewed before being merged. This could include an increased emphasis on peer reviews as well as improved test cases and automation pipelines to catch issues early.

Outside of the backend server, the Swift application and Privy recovery site components appear to be in a much healthier state of security. Once issues identified via this audit are remediated and test cases plus automation are expanded, we believe that LiquidMax will be in a position of strong security posture.

4.1. 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.