Zellic

Pyth Crosschain

Smart Contract Security Assessment

October 24, 2022

Prepared for:

Pyth

Prepared by:

Filippo Cremonese and Jasraj Bedi

Zellic Inc.

Contents

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please email us at hello@zellic.io or contact us on Telegram at https://t.me/zellic_io.

1 Executive Summary

Zellic conducted an audit for Pyth from October 18th to October 24th, 2022. This was re-audit of the codebase after our first audit in May 2022.

Our general overview of the code is that it was very well-organized and structured. The code coverage is high, and tests are included for the majority of the functionality. The comments and documentation are adequate and include warnings about which best practices should be followed to use the project correctly. The code is easy to comprehend.

We applaud Pyth for their attention to detail and diligence in maintaining incredibly high code quality standards in the development of Pyth Crosschain.

Zellic thoroughly reviewed the Pyth Crosschain codebase to find protocol-breaking bugs as defined by the documentation and to find any technical issues outlined in the Methodology section (2.2) of this document.

Specifically, taking into account Pyth Crosschain's threat model, we focused heavily on issues that would allow to manipulate price feeds, cause a denial of service, or cause unauthorized governance actions.

During our assessment on the scoped Pyth Crosschain contracts we discovered no security issues. A few minor aesthetic observations are recorded for Pyth's benefit in the Discussion section (4) at the end of the document.

2 Introduction

2.1 About Pyth Crosschain

Pyth is a first party financial oracle with real-time market data on-chain. It aims to bring valuable financial market data to DeFi applications and the general public. Being native to Solana, the prices can only be read by clients on the same network. Pyth leverages the cross-chain arbitrary messaging capabilities of Wormhole to bridge the price data to other chains, such as Ethereum and Terra.

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 open-source 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. We analyze the scoped smart contract code using automated tools to quickly sieve out and catch these shallow bugs. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so forth 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 manually review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents. We also thoroughly examine the specifications and designs themselves for inconsistencies, flaws, and vulnerabilities. This involves use cases that open the opportunity for abuse, such as flawed tokenomics or share pricing, arbitrage opportunities, and so forth.

Complex integration risks. Several high-profile 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. We perform a meticulous review of all of the contract's possible external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so forth.

Code maturity. We review for possible 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, upgradeability weaknesses, centralization risks, and so forth.

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; we assign it on a case-by-case basis based on our professional judgment and experience. As one would expect, both the severity and likelihood of an issue affect its impact; for instance, a highly severe issue's impact may be attenuated by a very low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Similarly, Zellic organizes its reports such that the most important findings come first in the document rather than being 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 numerous soft factors, such as our clients' threat models, their business needs, their project timelines, and so forth. We aim to provide useful and actionable advice to our partners that consider their long-term goals rather than simply provide a list of security issues at present.

2.3 Scope

The engagement involved a review of the following targets:

Pyth Crosschain Contracts

Repository	https://github.com/pyth-network/pyth-crosschain/
Versions	`079f8cc0b7ebcdd19415547269c4187fd91b8e70`
Contracts	• `ethereum/contracts/pyth/Pyth.sol` • `ethereum/contracts/pyth/PythGetters.sol` • `ethereum/contracts/pyth/PythGovernance.sol` • `ethereum/contracts/pyth/PythGovernanceInstructions.sol` • `ethereum/contracts/pyth/PythInternalStructs.sol` • `ethereum/contracts/pyth/PythSetters.sol` • `ethereum/contracts/pyth/PythState.sol` • `ethereum/contracts/pyth/PythUpgradable.sol` • `solana/pyth2wormhole/program/src/migrate.rs` • `solana/pyth2wormhole/program/src/set_is_active.rs`
Type	Solidity, Rust
Platform	EVM-compatible, Solana

2.4 Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of two person-weeks. The assessment was conducted over the course of one calendar week.

Contact Information

The following project managers were associated with the engagement:

Filippo Cremonese, Engineer
fcremo@zellic.io

Jasraj Bedi, Engineer, Co-founder
jazzy@zellic.io

The following consultants were engaged to conduct the assessment:

Filippo Cremonese, Engineer
fcremo@zellic.io

Jasraj Bedi, Engineer, Co-founder
jazzy@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

October 18, 2022	Start of primary review period
October 24, 2022	End of primary review period

3 Detailed Findings

Zellic found no security issues in this audit.

4 Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment.

4.1 Misleading error

`InvalidSigner` is returned by Pyth2Wormhole `migrate` function if the wrong account is provided as the system program:

```
if *accs.system_program.key ≠ system_program::id() {
    trace!(
        “Invalid system program, expected {:?}), found {}”,
        system_program::id(),
        accs.system_program.key
    );
    return Err(SolitaireError::InvalidSigner(
        accs.system_program.key.clone(),
    ));
}
```

The name of this error is misleading, and ideally a more aptly named error should be returned.

4.2 Review of pull request #355

While the audit was ongoing we agreed to broaden the scope to review pull request #355, based on commit `b05a63f` and including the range of commits from `203b22b` to `5924755`. We found no security issues in the pull request. We have two minor aesthetic issues to report.

The public function `AuthorizeGovernanceDataSourceTransfer` in `PythGovernance.sol` does not follow the style of the rest of the codebase, which is to camel case function names with the first letter being lowercase.

Furthermore, the comment documenting function `parseRequestGovernanceDataSourceTransferPayload` seems to be copy-pasted from function `parseAuthorizeGovernanceDataSourceTransferPayload`.

5 Audit Results

Zellic did not discover any security issues in the code.

5.1 Disclaimers

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 additional code added to the assessed project after the audit version of 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 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.

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.