QuillAudits

AUDIT REPORT

February, 2025

For

POLY FUND

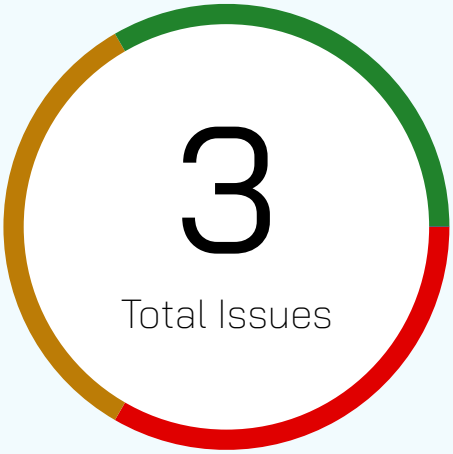
Table of Content

Executive Summary

Project name	Poly Fund
Overview	Poly Fund represents a vault-based prediction system where users deposit USDC in exchange for shares. The manager places bets using the vault balance, and the exchange validates and processes the bet using ERC-1155 tokens. After the prediction event concludes, users can redeem shares for USDC based on the outcome.
Timeline	31st January 2025 - 10 February 2025
Project URL	polyfund.so
Audit Scope	The scope of this Audit was to analyze the Poly Fund Smart Contracts for quality, security, and correctness. https://github.com/ash24r/poly-vaults-contract/tree/master/contracts
Contracts in Scope	PolyVaultFactory.sol PolyVault.sol ERC4626Fees.sol
Commit Hash	2eb4d16a8c58b798a3d1abd20acd853c6edc81ec
Language	Solidity
Blockchain	Polygon
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	31st January 2025 - 10 February 2025
Updated Code Received	24th February 2025

Review 2	25th Febriary 2025 - 3rd March 2025
Fixed In	9b26f4486267dbd2e991690f6141e7e107caf0ac

Number of Issues per Severity



High	1 (33.33%)
Medium	1 (33.33%)
Low	1 (33.33%)
Informational	0 (0.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	**1**	**1**	0	0
Acknowledged	0	0	**1**	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ✓ Access Management
- ✓ Arbitrary write to storage
- ✓ Centralization of control
- ✓ Ether theft
- ✓ Improper or missing events
- ✓ Logical issues and flaws
- ✓ Arithmetic Computations Correctness
- ✓ Race conditions/front running
- ✓ SWC Registry
- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC's conformance
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Multiple Sends
- ✓ Using suicide
- ✓ Using delegatecall
- ✓ Upgradeable safety
- ✓ Using throw

- [x] Using inline assembly
- [x] Unsafe type inference
- [x] Style guide violation
- [x] Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools And Platforms Used For Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity statistical analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

Security vulnerabilities identified that must be resolved and are currently unresolved.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

High Severity Issues

Reward Distribution Front-Running by Attacker via Increased Holdings

Resolved

Path

PolyVault.sol , ERC4626.sol

Function

deposit() , redeemPositions()

Description

An attacker can make an immediate profit by calling a deposit just before calling redeemPositions() of the vault. Since anyone can use this function to increase the total assets in the vault, the attacker could then call to withdraw immediately within a single transaction.

An attacker exploits the reward redemption mechanism in a vault contract:
1. Initial state: 100 ETH total assets (TA), 100 shares outstanding
2. Pending rewards: 50 ETH unclaimed
3. Attack flow in one transaction:
- Attacker deposits 100 ETH => gets 100 shares
- Calls redeemPositions() => claims 50 ETH rewards into a vault
- New TA = 250 ETH, total shares = 200, share price = 1.25 ETH
- Attacker withdraws 100 shares => receives 125 ETH
- Profit: 25 ETH stolen from existing shareholders

Recommendation

Set a 1-day cooldown period before the deposit.

Medium Severity Issues

First Depositor can inflate Vault

Resolved

Path

PolyVaultFactory.sol

Function

createVault()

Description

The Vault contract uses `_decimalsOffset` to add more precision to share values. The issue is that the value of the `_decimalsOffset` would be 0 if the underlying token had 6 decimals (which is the case for USDC).

So share calculation would be:

=> assets.mulDiv(newTotalSupply + 1, newTotalAssets + 1, rounding)

Attack scenario:
1. Attacker deposits 10 wei worth of USDC and receives 10 shares (total shares in the vault: 10 + 1 virtual share).
2. He manually transfers 11,000 USDC (vault balance = 11,000 + 10 wei).
3. Now, Alice deposits 1,000 USDC but mint 0 share.
4. The total shares become 11, and the vault balance is 12,000 + 10 wei (1,090 USDC per share).
5. Attacker redeems 10 shares and receives 1,090 × 10 = 10,909.09 USDC.

Outcome:
Attacker's Loss: 100 USDC
Alice's Loss: 1,000 USDC

Recommendation

Create "dead shares" like it's done in UniswapV2/Balancer, i.e. deposit the initial amount to the vault on the initialization step.

Low Severity Issues

Gas Dos possible

Acknowledged

Path

PolyVault.sol

Function

multiRedeemShares()

Description

If the ctfTokenIds array grows too large (e.g., due to many ERC1155 tokens being sent to the vault), the loop will consume an excessive amount of gas. This can cause the function to revert, effectively denying service to users who want to redeem their shares.Users may be unable to redeem their shares if the ctfTokenIds array becomes too large.

Recommendation

Limit the Size of the ctfTokenIds Array

Functional Tests

Some of the tests performed are mentioned below:

- isValidSignature function performs as expected and does not require a nonce update since the Polymarket contract manages it
- deposit function operates flawlessly with no rounding issues
- The fee mechanism works as expected
- Withdraw function works as expected
- The manager creates the vault and deploys it correctly
- Overall all contracts are permissionless
- There is no pause mechanism to stop malicious activity in the protocol

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Poly Fund. We performed our audit according to the procedure described above.

Issues of High,medium and low severity was found. In the end, Poly Fund team resolved high and medium issue and acknowledged low issue

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.
Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

February, 2025

For

POLY FUND

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com