QuillAudits

Audit Report
May, 2023

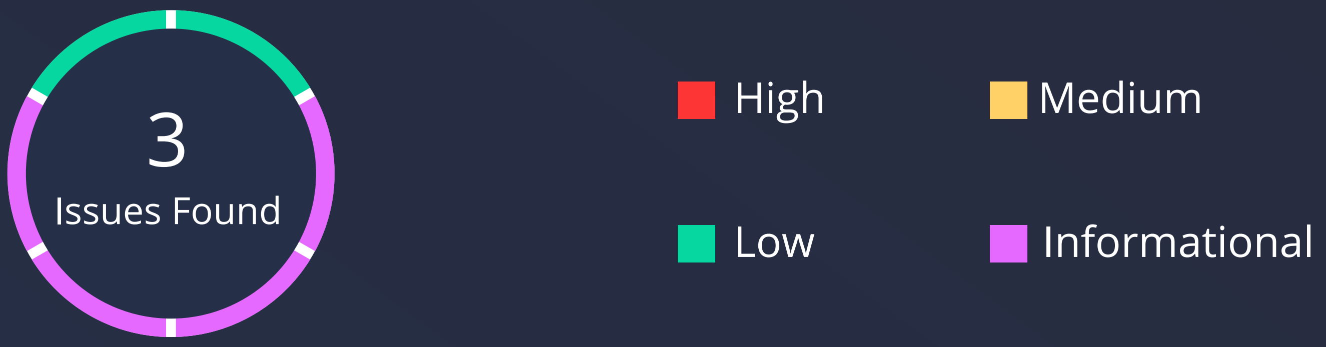
For

xETH

Table of Content

Executive Summary

Project Name	xETH
Overview	xETH contracts leverage on providing a liquidity pool for its token and stETH on the Curve plain pool and staking the derived CRV token on Convex Protocol in order to benefit from its reward system. The Algorithmic Market Operation (AMO) contract automatically regulates the supply and demand of the xETH token. The contract allows the default admin to provide liquidity of xETH/stETH tokens into the xETH/stETH pool and created with an rebalancer mechanism that relies on the quote of an offchain rebalance defender or the contract quote. CRV tokens derived from providing liquidity are afterwards staked on the Convex protocol. The wxETH contract allows for xETH token holders to stake their xETH for a period of time in order to earn dripped xETH tokens. On staking the xETH tokens, the wxETH token and on withdrawing, the wxETH token are burned and the xETH staked with rewards from the dripping activity can be withdrawn.
Timeline	13 April, 2023 - 3 May, 2023.
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse the following Xeth's codebases for quality, security, and correctness: xETH, WrappedXETH, CVXStaker and xETH_AMO. *https://gitlab.com/zrx_hilbert/xeth/-/tree/main/src* **Commit hash:** 6f53f1faffcb0baaf26f78f3e77d8aa26e9ee510
Fixed In	*https://gitlab.com/zrx_hilbert/xeth/-/tree/main/src* **Commit hash:** 6a2a086ad064a59b2d1ee76507d09ad5fe3e87de



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

Types of Severities

High

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

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

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

Informational

These are 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
These are the issues identified in the initial audit and have been successfully fixed.

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.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

Techniques and Methods

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

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token 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 analysed 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

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 analysed, 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 behaviour 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, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Common Issues

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

A.1 Floating Solidity (pragma solidity ^0.8.0)

Description

Both contracts have a floating solidity pragma version. This is also present in inherited contracts. Locking the pragma helps to ensure that the contract does not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. The recent solidity pragma version also possesses its own unique bugs.

Remediation

Making the contract use a stable solidity pragma version prevents bugs that could be ushered in by prospective versions. Using a fixed solidity pragma version while deploying is recommended to avoid deployment with versions that could expose the contract to attack.

Status

Resolved

B. xETH.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

No issues found

C. wxETH.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

No issues found

D. CVXStaker.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

D.1 Missing Input Validation

Description

Some critical functions that allow admin to set critical state variables do not have sanity checks to prevent setting these variables to an address zero or invalid input that could cause other function calls to fail if not set to proper addresses.
These functions include:
- Constructor
- setCvxPoolInfo
- setOperator
- setRewardRecipient

Remediation

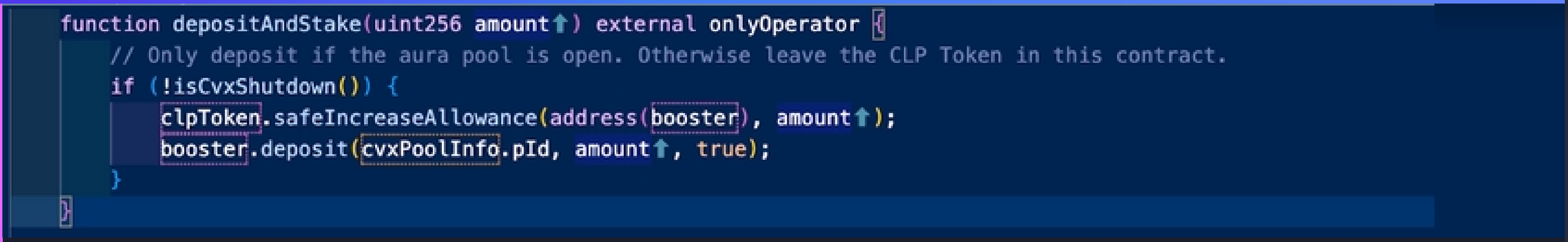
Add a validation check in these functions to prevent inaccurate assignment of values.

Status

Resolved

Informational Issues

D.2 Inconsistency with Comment and Code Implementation



```
function depositAndStake(uint256 amount) external onlyOperator {
    // Only deposit if the aura pool is open. Otherwise leave the CLP Token in this contract.
    if (!isCvxShutdown()) {
        clpToken.safeIncreaseAllowance(address(booster), amount);
        booster.deposit(cvxPoolInfo.pId, amount, true);
    }
}
```

Description

With the contract designed to interact with the Convex protocol, the function above interacts directly with the Convex Booster supposing that the convex loop is open but the comment in the contract says otherwise.

Remediation

Correct code comment to be consistent with contract implementation.

Status

Acknowledged

E. xETH_AMO.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

No issues found

General Recommendation

As an algorithmic market operation, when the amount of xETH token in the pool reaches some certain threshold, the rebalance up and down mechanism ensures the burning and minting of xETH tokens based on the quote from the offchain rebalance defender or the quote generated within the contract. If this offchain rebalance defender address fails at any point, the rebalancing mechanism is halted. Team must ensure to be watchful of this external offchain property.

Functional Testing

Some of the tests performed are mentioned below

xETH.sol

- ✓ Should get the name of the token
- ✓ Should get the symbol of the token
- ✓ Should be able to set new addresses as the minter while revoking the previous minter role if any.
- ✓ Should assign and allow the AMO contract mint xETH tokens when adding liquidity to the pool.
- ✓ Should allow the minter to also burn the xETH token when removing liquidity.
- ✓ Should disallow the transfer of tokens when the contract is paused.
- ✓ Should allow only assigned the address with the Pauser role to pause/unpause the contract.

wxETH.sol

- ✓ Should preview the exchange rate of xETH to wxETH before staking into the contract and vice versa.
- ✓ Should let the contract owner start and stop dripping activity in the contract.
- ✓ Should allow the contract owner to set the drip rate.
- ✓ Should confirm the xETH token provided by the contract owner with the addLockedFunds function.
- ✓ Should allow xETH token holders to successfully stake into the contract and mints a proportion of wxETH.
- ✓ Should allow wxETH token holders to successfully unstake into the contract and mints a proportion of xETH.

CVXStaker.sol

- ✓ Should set the convex pool info in order to stake the CRV token successfully.
- ✓ Should successfully deposit into the Convex Booster when addLiquidity is called by the operator.
- ✓ Should withdraw from Convex Base Pool the CRV token when liquidity is to be removed.
- ✓ Should allow only the owner to withdraw CRV tokens straight into the CVXStaker contract with the withdrawAllAndUnwrap.
- ✓ Should know the staked balance and earned rewards in the Convex protocol.

AMO.sol

- ✓ Should allow the bot successfully rebalance up or rebalance down the xETH token in relation to the pool.
- ✓ Should check if the cool down period has passed when rebalancing up or down.
- ✓ Should allow the default admin add liquidity of xETH/stETH into the whitelisted Curve pool.
- ✓ Should allow the default admin to remove liquidity from the pool.
- ✓ Should allow the default admin to add and remove only stETH tokens from the pool.

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 xETH. We performed our audit according to the procedure described above.

Some issues low and informational severity were found, Some suggestions
and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the xETH Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the xETH Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

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.

850+
Audits Completed

$16B
Secured

800K
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2023

For

xETH

QuillAudits

Canada, India, Singapore, United Kingdom

www.quillaudits.com

audits@quillhash.com