#HashEx
BLOCKCHAIN SECURITY

Crystl Vaults

smart contracts
preliminary audit report
for internal use only

March 2022

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Crystl Finance team to perform an audit of their smart contracts. The audit was conducted between 2022-01-18 and 2022-01-28.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at @polycrystal/polycrystal-vaults GitHub repository after 426d351 commit. The updated code was rechecked in the d5091b0 commit.

2.1 Summary

Project name	Crystl Vaults
URL	https://www.polycrystal.finance
Platform	Polygon Network
Language	Solidity

2.2 Contracts

Name	Address
ContractSizer	
BaseStrategySwapLogic	
VaultHealerBase	
VaultHealerEarn	
VaultHealerGate	
Magnetite	
VaultFeeManager	
Fee	
BaseStrategyVaultHealer	
VaultHealerFactory	
BoostPool	
BaseStrategy	
TacticMasterHealerWithReferral	
Cavendish	
VHStrategyProxy	
VaultHealer	
StrategyVHStandard	
TacticMasterHealer	

StrategyConfig

VaultHealerBoostedPools

QuartzUniV2Zap

PrismLibrary

VaultView

Tactics

TacticMiniApe

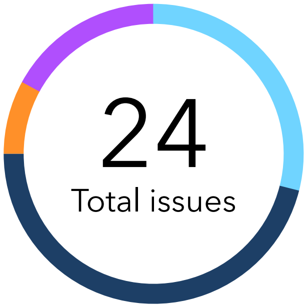
LibQuartz

StrategyQuick

VaultHealerAuth

PRBMath

3. Found issues



●	High	2 (8%)
●	Medium	4 (17%)
●	Low	7 (29%)
●	Info	11 (46%)

C1. ContractSizer

ID	Severity	Title	Status
C1-01	● Info	extcodesize may return 0 for a contract	✓ Resolved

C2. BaseStrategySwapLogic

ID	Severity	Title	Status
C2-01	● Medium	Wrong slippage and deadline usage	✓ Acknowledged
C2-02	● Low	Same dust parameter for tokens with different decimals	✓ Resolved
C2-03	● Low	External calls may spent gas	✓ Resolved
C2-04	● Info	Redundant code	✓ Resolved
C2-05	● Info	Inconsistent comment	✓ Resolved
C2-06	● Info	Code style	✓ Resolved

C3. VaultHealerBase

ID	Severity	Title	Status
C3-01	● Low	Redundant code	✓ Resolved
C3-02	● Info	Lack of events	✓ Resolved
C3-03	● Info	No revert message	✓ Resolved

C4. VaultHealerEarn

ID	Severity	Title	Status
C4-01	● Low	Gas optimization	✓ Resolved

C5. VaultHealerGate

ID	Severity	Title	Status
C5-01	● High	accRewardTokensPerShare monotonic growth	✓ Resolved
C5-02	● Medium	lastEarnBlock is not updated	✓ Resolved
C5-03	● Low	Division error	✓ Resolved
C5-04	● Info	Code style	✓ Resolved

C6. Magnetite

ID	Severity	Title	Status
C6-01	● Medium	Constant return	✓ Resolved
C6-02	● Low	Can't be called externally	✓ Resolved
C6-03	● Info	Sub-paths can only be updated manually	✓ Resolved

C7. VaultFeeManager

ID	Severity	Title	Status
C7-01	● High	Exaggerated owner rights	✓ Resolved
C7-02	● Low	Gas optimization	✓ Resolved

C8. Fee

ID	Severity	Title	Status
C8-01	● Info	Total fees may reach 100%	✓ Acknowledged

C9. BaseStrategyVaultHealer

ID	Severity	Title	Status
C9-01	● Info	Code style	✓ Resolved

C10. VaultHealerFactory

ID	Severity	Title	Status
C10-01	● Info	Code style	✓ Resolved

C11. BoostPool

ID	Severity	Title	Status
C11-01	● Medium	Logic violation	✓ Resolved

4. Contracts

C1. ContractSizer

Overview

The contract consist of only one function `sizeOf()`, that returns the passed contract's code size.

Issues

C1-01 extcodesize may return 0 for a contract ● Info ✓ Resolved

The size check can be bypassed [1].

Team response

No action needed, the contract was only created for ad hoc use.

C2. BaseStrategySwapLogic

Overview

The abstract contract and the part of the StrategyVHStandard contract inheritance scheme.

Issues

C2-01 Wrong slippage and deadline usage ● Medium ✓ Acknowledged

Calculating slippage and setting a deadline with `block.timestamp` inside the `safeSwap()` function will not affect transaction success. These parameters should be obtained off-chain. Calculating slippage as a percentage of `router.getAmountsOut()` effects only the transfers of tokens with fees on transfers.

This issue was moved to the BaseStrategy contract in the code update.

Team response

We're happy that the risk is not substantial given that we only swap on earn, and are not at a size where front-running would be that profitable for anyone. Setting params offchain too much of a lift right now. Might look at using Uniswap priceCumulativeLast solution at some point in the future.

C2-02 Same dust parameter for tokens with different decimals ● Low ✓ Resolved

The value of a token may differ according to the `totalSupply` and `decimals` value. Fixing the significance threshold is incorrect as the same amount of different tokens can be valued differently.

Team response

Our plan here is to create separate dust variables for each token - we will reduce these to one byte variables to save on storage space

C2-03 External calls may spent gas ● Low ✓ Resolved

Transfers of native currency with `receiver.call{value}` in L111 may cause significant gas spending if the receiver implements the fallback function. This can be avoided by fixing the call gas to a constant or variable amount.

C2-04 Redundant code ● Info ✓ Resolved

L34 is unnecessary and could be removed.

C2-05 Inconsistent comment

● Info ✓ Resolved

The correctness of the `for()` loop execution depends on the last element in the `earned` array. If it's not zero-initialized, a transaction would fail due to an indexation error, whereas the problem is in the last array element value.

C2-06 Code style

● Info ✓ Resolved

TODO & testing comments in L59, 82, 123, 126, 132, 134-147, 152, 160, 164, 170, 174..

C3. VaultHealerBase

Overview

The abstract contract, the part of the VaultHealer contract inheritance scheme.

Issues

C3-01 Redundant code

● Low ✓ Resolved

The `findVid()` function has internal visibility but is never used.

C3-02 Lack of events

● Info ✓ Resolved

When `vaultFeeManager` is changed no event is emitted.

C3-03 No revert message

● Info ✓ Resolved

L92 requirement checks the vault id for inequality to zero. The opposite means that the passed strategy address is invalid. If the requirement is not met, the transaction is reverted with an empty message.

C4. VaultHealerEarn

Overview

The abstract contract, the part of the VaultHealer contract inheritance scheme.

Issues

C4-01 Gas optimization ● Low ✓ Resolved

a. Unchecked math may save gas in L110, L114.

b. Excessive reads from storage in L103, L105.

C5. VaultHealerGate

Overview

The abstract contract, the part of the VaultHealer contract inheritance scheme. It realizes `deposit()` and `withdraw()` functions through which users will interact with the VaultHealer.

Issues

C5-01 accRewardTokensPerShare monotonic growth ● High ✓ Resolved

The `accRewardTokensPerShare` variable stores the number of target tokens that may be received for the current strategy token. `accRewardTokensPerShare` is recalculated each time during deposit and withdrawal operation with vaults that have `targetVid`. In the `UpdatePoolAndRewarddebtOnDeposit()` and `UpdatePoolAndWithdrawCrystlOnWithdrawal()` functions updated variable is received by price accumulation, this results in a constant `accRewardTokensPerShare` growth. In other words, the token per token ratio only increases what breaks vaults' tokenomics and makes deposited tokens not fully backed by the declared target tokens amount. Token per token ratio accumulation may lead to flashloan attacks and users' assets loss.

Recommendation

Without the documentation, we can't validate the accRewardTokensPerShare logic. We recommend adding unit tests for the strategies with non-zero targetVid.

C5-02 lastEarnBlock is not updated

● Medium ✓ Resolved

lastEarnBlock field in VaultInfo structure is used in earn functions to ensure a vault is processed only once in a block. However, the field is updated neither in Strategy nor in VaultHealer methods. Thus, the vault may be earned multiple times in the block. The issue was detected in the update commit.

Recommendation

Consider update the variable after a successful call or remove the corresponding checks.

C5-03 Division error

● Low ✓ Resolved

There are possible division errors while targetVidShares calculation in withdrawTargetTokenAndUpdateOffsetsOnWithdrawal() function.

```
_vidSharesRemoved * (totalSupply(targetVid) + totalOffset) / totalSupply(_vid)
- fromOffset * _vidSharesRemoved / balanceOf(_from, _vid)
```

If totalSupply(_vid) or balanceOf(_from, _vid) is much larger than its numerator, one of the multipliers will be nullified and will lead to further targetVidShares miscalculation. This may cause withdrawal errors and users' funds loss. The issue was detected in the update commit.

Recommendation

Consider reducing fractions to a single one:

```
uint256 totalSupply_vid = totalSupply(_vid);
uint256 balanceOf_from_vid = balanceOf(_from, _vid);

( _vidSharesRemoved * (totalSupply(targetVid) + totalOffset) * balanceOf_from_vid
- fromOffset * _vidSharesRemoved * totalSupply_vid )
/ balanceOf_from_vid / totalSupply_vid
```

Take note that this approach may cause an overflow of numerator.

C5-04 Code style ● Info ✓ Resolved

TODO & testing comments in L32, 33, 35, 144, 154, 174, 176, 186, 200.

C6. Magnetite

Overview

The contract generates and stores swaps paths.

Issues

C6-01 Constant return ● Medium ✓ Resolved

`pathAuth()` always returns true making the `findAndSavePath()` function accessible for anyone.

Team response

We have reinstated the check within pathAuth(), and it is working fine again now (it was only removed originally for debugging, and our mistake not to reinstate it)

C6-02 Can't be called externally ● Low ✓ Resolved

The `setAutoPath_()` function has external visibility, but the L45 requirement allows calls only to `address(this)` contract. Thus, the function can only be called internally then visibility should be changed or access to the function should be reconsidered.

C6-03 Sub-paths can only be updated manually ● Info ✓ Resolved

During sub-path generation in `_setPath()`, intermediate paths that can be updated only manually with `overridePath()` are created.

C7. VaultFeeManager

Overview

The contract manages fee percents and fee receivers.

Issues

C7-01 Exaggerated owner rights ● High ✓ Resolved

The owner is able to increase earn and withdrawal fees up to 100% anytime. In this case, users' funds may be considered locked, since if they attempt to withdraw their deposits, they would be totally transferred to the owner.

Recommendation

The FEE_SETTER role must belong to a Timelock contract with a minimum delay of at least 24 hours. This won't stop the admin from possible tax shifts but will help users to be informed about upcoming changes. They will be able to make a decision in advance before the fee increases.

Team response

We have put in a limit of 30% on earn fee. We considered the timelock option, but in the end decided that because the fee is only on earn (not a withdrawal fee on principal, for example) that we're ok with this not being in a timelock.

C7-02 Gas optimization ● Low ✓ Resolved

FEE_MAX is declared but not used.

C8. Fee

Overview

Library defining Fee datatype.

Issues

C8-01 Total fees may reach 100% ● Info ✓ Acknowledged

The project taxes earns from masterchefs pools and all withdrawal operations. Each of these taxes can be up to 100%, resulting in the owner getting all masterchef rewards and all user's funds if one wants to withdraw deposits.

C9. BaseStrategyVaultHealer

Overview

The abstract contract, the part of the StrategyVHStandard contract inheritance scheme.

Issues

C9-01 Code style

● Info ✓ Resolved

TODO & testing comments in L13, 14, 15, 36, 37.

C10. VaultHealerFactory

Overview

The abstract contract, the part of the VaultHealer contract inheritance scheme. The contract includes gas-efficient functionality for adding new vaults using a proxy pattern.

Issues

C10-01 Code style

● Info ✓ Resolved

TODO & testing comments in L9, 28, 30-41.

C11. BoostPool

Overview

Contract for additional rewards distribution to be operated via VaultHealer.

Issues

C11-01 Logic violation ● Medium ✓ Resolved

When new `bonusEndBlock` is set in `setBonusEndBlock()` function, `rewardTotalSupply` and/or `rewardPerBlock` should be updated. Otherwise, pool rewards may exceed `totalSupply` of `rewardToken`.

C12. BaseStrategy

Overview

The abstract contract, the part of the StrategyVHStandard contract inheritance scheme. No issues were found.

C13. TacticMasterHealerWithReferral

Overview

The auxiliary contract for interacting with MasterChef-like contracts via DELEGATECALL. No issues were found.

C14. Cavendish

Overview

The library for strategy proxy deployment. Was added in the update. No issues were found.

C15. VHStrategyProxy

Overview

The proxy for StrategyVHStandard. Created architecture is gas-efficient solution for deployment of strategy contracts. The proxy also has rights delimitation and provides only view access to strategy's methods if caller is not VaultHealer. No issues were found.

C16. VaultHealer

Overview

Main contract in the VaultHealer scheme. After the update the contract overrides isApprovedForAll() function approving all deposit proof NFTs for trustedForwarder. This means, trustedForwarder is able to transfer users' NFTs to himself and withdraw their deposits. No issues were found.

C17. StrategyVHStandard

Overview

Main strategy contract. No issues were found.

C18. TacticMasterHealer

Overview

Auxiliary contract for interacting with MasterChef-like contracts via DELEGATECALL. No issues were found.

C19. StrategyConfig

Overview

The library helps to extract information from Tactics data structures. Was added in the update. No issues were found.

C20. VaultHealerBoostedPools

Overview

The abstract contract, the part of the VaultHealer contract inheritance scheme. It allows the owner or anyone with `BOOST_ADMIN` role to manage the boosting pools. No issues were found.

C21. QuartzUniV2Zap

Overview

The contract helps with the swap operation of vaults tokens. No issues were found.

C22. PrismLibrary

Overview

A generic UniswapV2 library. Was added in the update. No issues were found.

C23. VaultView

Overview

Support contract for VaultHealer. Contains view functions of the VaultHealer to reduce the size of its deployed bytecode. No issues were found.

C24. Tactics

Overview

The library helps Tactics to interact with masterchefs. Was added in the update. No issues were found.

C25. TacticMiniApe

Overview

Auxiliary contract for interacting with MasterChef-like contracts via DELEGATECALL. No issues were found.

C26. LibQuartz

Overview

The library with auxiliary swap functions. Was added in the update. No issues were found.

C27. StrategyQuick

Overview

Strtegy extension for the QuickSwap's DragonLair. Was introduced with the update. No issues were found.

C28. VaultHealerAuth

Overview

Authentication contract for VaultHealer, based on the AccessControlEnumerable model from OpenZeppelin library. Was introduced with the update. No issues were found.

C29. PRBMath

Overview

Fixed-point math library. Was introduced with the update. No issues were found.

5. Conclusion

2 high severity issues were found during the audit and were fixed in the update afterwards. The reviewed contracts are highly dependent on the owner's account. The contracts are upgradeable and their implementation can be switched by the owner.

Appendix A. Issues severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Info.** Issues that do not impact the contract operation. Usually, info severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

Appendix C. References

1. Bypass Contract Size Check. URL: https://solidity-by-example.org/hacks/contract-size.

✉ contact@hashex.org

@hashexbot

blog.hashex.org

linkedin

github

twitter

HashEx
BLOCKCHAIN SECURITY