Guard

Smart contracts security assessment

Final report

Tariff: Top

Champion Optimizer

January 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Champion Optimizer**.

Two contracts of Champion Optimizer protocol were audited.

The first smart contract, named `ChampionOptimizerVaultV1`, appears to be a base contract for managing a specific investment strategy. It contains functions for handling deposits, withdrawals, and the distribution of fees to various parties.

The second smart contract, named `StrategySolidlyGaugeLPThena`, appears to be a specific implementation of the strategy manager contract, using a liquidity pool token as the primary investment. The contract's main functionality is to deposit and withdraw funds and contains a mechanism for handling and distributing fees. Also, the contract has a function `harvest()` for any user to harvest rewards. The caller of this function will get a percentage of tips for calling this function.

The code is available at the GitHub repository and was audited after the commit 0eee2e30879b545466506bbfc9a8fdb0292413cf.

Report update.

The contracts' code was updated according to this report and rechecked after the commit fa3b2d5d48a65be22c1144ee32d906b1ce1dd0d1.

Name	Champion Optimizer
Audit date	2023-01-22 - 2023-01-27
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
ChampionOptimizerVaultV1	
StrategySolidlyGaugeLPThena	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed

Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Strategy update (ChampionOptimizerVaultV1)
Status: Fixed

In case the strategy variable is updated, users won't be able to return their funds, and the owner will have access to them.

2. Change of uniRouter and vault (StrategySolidlyGaugeLPThena)
Status: Fixed

The owner can change variables like `uniRouter` and `vault`. Changing them to malicious ones will lead to the loss of users' funds.

Recommendation: Delete functions `setUniRouter()` and `setVault()`.

3. Front-run vulnerability (StrategySolidlyGaugeLPThena)
Status: Partially fixed

The contract is vulnerable to sandwich attacks or attacks on pools using flash loans.

Recommendation: Reconsider the logic of swaps with adding liquidity.

Update: The possibility of attacks using flash loans was eliminated in the commit

fa3b2d5d48a65be22c1144ee32d906b1ce1dd0d1, but there is a possibility of sandwich attacks, modifier `gasThrottle` won't help because such things as mev bots can bundle transactions.

Medium severity issues

1. Approvals for a new router (StrategySolidlyGaugeLPThena)
Status: Fixed

When the owner changes the router, the contract needs to make new approvals for the new router, but it doesn’t happen.

2. Problems in addLiquidity() function (StrategySolidlyGaugeLPThena)
Status: Fixed

In the function `addLiquidity()`, edge cases when `lpToken0==output` or `lpToken1==output` aren’t handled in the first if.

Also, on lines 210 and 215, there are errors. Variable `outputToLp0Route.length` should be replaced with `outputToLp0Route.length-1`.

Low severity issues

1. Not enough events (ChampionOptimizerVaultV1)
Status: Fixed

Low amount of events. We recommend emitting events on important value changes to simplify tracking them off-chain.

2. Gas optimisation (ChampionOptimizerVaultV1)
Status: Partially fixed

1. Function `want()` can be replaced by a variable that is assigned to a value in the constructor, and the function `upgradeStrat()`
2. In the function `earn()` global variable `strategy` is read multiple times
3. In the function `withdraw()` global variable `want` is read multiple times

4. In the function `upgradeStrat()` global variable `stratCandidate.implementation` is read multiple times
5. Global variable `want` can be immutable

Update: Points 1, 2, and 4 in the list were closed in the commit fa3b2d5d48a65be22c1144ee32d906b1ce1dd0d1.

3. Check of approvalDelay (ChampionOptimizerVaultV1)
Status: Fixed

Variable `approvalDelay` should be big enough. In the constructor, there are no checks on that.

4. Gas optimization (StrategySolidlyGaugeLPThena)
Status: Partially fixed

1. In the constructor, a global variable `outputToNativeRoute.length` is read multiple times
2. In the constructor, global variables `want`, `lpToken0`, `lpToken1`, and `output` are read after write
3. In the modifier `gasThrottle()` global variable `gasprice` is read multiple times
4. Global variables `native`, `output`, `want`, `lpToken0`, `lpToken1`, `gauge`, `gasprice`, `stable`, `lp0Decimals`, `lp1Decimals` can be marked as immutable
5. In the function `withdraw()`, global variables `want` and `vault` are read multiple times
6. In the function `chargeFees()` global variable `native` is read multiple times
7. In the function `addLiquidity()` global variables `output`, `uniRouter`, `lpToken0`, `lpToken1`, `outputToLp0Route`, `outputToLp1Route`, and `stable` are read multiple times
8. In the function `_giveAllowances()`, global variables `uniRouter`, `lpToken0`, and `lpToken1` are read multiple times
9. In the function `_giveAllowances()`, there is no need to approve at first zero allowance

Update: Points 1-6 and 8-9 in the list were closed in the commit fa3b2d5d48a65be22c1144ee32d906b1ce1dd0d1.

5. Not used variables (StrategySolidlyGaugeLPThena)
Status: Fixed

Variables `lastHarvest` and `rewards` aren't used anywhere.

6. Return values aren't checked (StrategySolidlyGaugeLPThena)
Status: Open

In the constructor results of calls aren’t checked.

7. Problem with gasThrottle (StrategySolidlyGaugeLPThena)
Status: Open

Modifier `gasThrottle` isn’t working because using the function `deposit()` in the contract ChampionOptimizerVaultV1, a user can bypass this restriction.

8. Not enough events (StrategySolidlyGaugeLPThena)
Status: Partially fixed

Low amount of events. We recommend emitting events on important value changes to simplify tracking them off-chain.

Conclusion

Champion Optimizer ChampionOptimizerVaultV1, StrategySolidlyGaugeLPThena contracts were audited. 3 high, 2 medium, 8 low severity issues were found.
2 high, 2 medium, 3 low severity issues have been fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

0x Guard