DeHacker

Code Security Assessment

CINDEX FINANCE

September 5th, 2024

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	CINDEX FINANCE
Platform	Ethereum
Website	cindex.finance/
Type	DeFi
Language	Solidity
Codebase	https://github.com/Cindex-finance/core

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	1	0	1
Medium	3	0	0	0	2	1
Minor	3	0	0	2	0	2
Informational	2	0	0	0	0	2
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
VCC	Vault.sol	14eb876859301d3d094fb0e6adbcc25590a2e1ec30677deffbadfec99bcfdfe5

Findings

ID	Title	Severity	Status
VAU-02	Updating Router Centralization RiskIn Vault.sol	Major	Acknowledged
VCC-01	Pausing Centralization Risk InVault.sol	Major	Resolved
CSC-01	Vulnerability In CindexSwap ContractDue To User-Controlled External Calls	Medium	Partially Resolved
VCC-02	Missing Validation OnlatestRoundData()	Medium	Resolved
VCC-07	Lack Of Input Validation In deposit()And withdraw()	Medium	Partially Resolved
CCB-01	Unsafe Integer Cast	Minor	Resolved
GLOBAL-01	Third-Party Dependency Usage	Minor	Acknowledged
VCC-04	Missing Zero Address Validation	Minor	Resolved
VCC-05	Potential Divide By Zero	Informational	Resolved
VCC-06	Potential Reversion In Protocol FeeCalculation	Informational	Resolved

MAJOR

VAU-02| UPDATING ROUTER CENTRALIZATION RISK IN Vault.sol

Issue	Severity	Location	Status
Centralization	Major	Vault.sol (v2): 100	Acknowledged

Description

In the contract Vault the role _owner has authority over the function shown in the diagram below. Any compromise to the _owner account may allow the hacker to take advantage of this authority and update the ICindexSwap router . The router is used for swapping tokens. If a swap is necessary, the Vault contract would transfer tokens to the router for swapping, and then receive the swapped tokens back from the router . If the router is compromised by a hacker, severedamage could occur, such as the loss of the swapping tokens.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Recommendation

Short Term:

Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private keycompromised;
- AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience

Permanent:

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles.
- OR
- Remove the risky functionality.

MAJOR

VCC-01 | PAUSING CENTRALIZATION RISK IN Vault.sol

Issue	Severity	Location	Status
Centralization	Major	Vault.sol (v1): 334, 338	Acknowledged

Description

In the contract Vault the role _owner has authority over the functions shown in the diagram below. Any compromise to the _owner account may allow the hacker to take advantage of this authority and pause/unpause the contract.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations; AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised.

Recommendation

AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality

MEDIUM

CSC-01 | VULNERABILITY IN CindexSwap CONTRACT DUE TOUSER-CONTROLLED EXTERNAL CALLS

Issue	Severity	Location	Status
Logical Issue	Medium	exchange/CindexSwap.sol (v1): 9	Partially Resolved

Description

The swap() function in the CindexSwap contract exhibits a critical security vulnerability by allowing external calls to anyaddress with data supplied by the user. data.extRouter and data.extCalldata are passed by the user and there is notany sort of validation performed on these two parameter.This design flaw could be exploited by a malicious actor to perform unauthorized transactions. For instance, if a user likeAlice has approved the CindexSwap contract to spend her USDT tokens, an attacker could manipulate the contract to callthe USDT token contract and transfer Alice's tokens to themselves

```
contract CindexSwap is ICindexSwap, OneInchRouterHelper {

      using Address for address;

      function swap(
            address tokenIn,
            uint256 amountIn,
            SwapData calldata data
       ) external override payable {
            TransferHelper.safeApproveInf(tokenIn, data.extRouter);
            data.extRouter.functionCallWithValue(
                  data.needScale ? _getOneInchInputData(data.extCalldata, amountIn) :
data.extCalldata,
                  tokenIn == TransferHelper.NATIVE ? amountIn : 0
            );
      }

      receive() external payable
{
```

Recommendation

It is recommended to restrict the swap() function's ability to make arbitrary external calls.

MEDIUM

VCC-02| MISSING VALIDATION ON latestRoundData()

Issue	Severity	Location	Status
Logical Issue	Medium	Vault.sol (v1): 316, 323	Resolved

Description

The functions queryAssetValue() and getPrices() in the Vault contract calls latestRoundData() from Chainlink toacquire the token's price. This function does not contain the checks to verify the price data hasn’t become outdated or stale.This could result in inaccurate calculations of values per share.

Recommendation

We recommend adding a validation to the return values of latestRoundData() to make sure that the price is not stale.
Reference: https://github.com/code-423n4/2021-08-notional-findings/issues/92.

MEDIUM

VCC-07| LACK OF INPUT VALIDATION IN deposit() ANDwithdraw()

Issue	Severity	Location	Status
Volatile Code	Medium	Vault.sol (v1): 109, 134, 155	Partially Resolved

Description

The Vault contract's deposit() and withdraw() functions pose a significant security risk by not implementingnecessary restrictions on the swapData parameter within the params struct. This parameter is crucial as it may be utilized to perform external calls to the router using the provided swapData . Allowing arbitrary and unchecked values in swapData could potentially endanger the contract, enabling execution of unintended or harmful operations.

```
Struct SwapData {
      address extRouter;
      bytes extCalldata;
      bool needScale;
}
```

Recommendation

It is recommended to implement checks and validations on the swapData parameter.

MINOR

CCB-01| UNSAFE INTEGER CAST

Issue	Severity	Location	Status
Incorrect Calculation	Minor	Vault.sol (v2): 326; Vault.sol (v1): 308, 319, 324	Resolved

Description

Type casting refers to changing an variable of one data type into another. The code contains an unsafe cast between integertypes, which may result in unexpected truncation or sign flipping of the value.

```
308 return uint256(price) * amount / (10 ** decimals);
```

```
319 prices[0] = uint256(price) * exchangeRate / PRECISION;
```

```
324 prices[1] = uint256(price2);
```

Recommendation

It is recommended to check the bounds of integer values before casting. Alternatively, consider using the SafeCast libraryfrom OpenZeppelin to perform safe type casting and prevent undesired behavior.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/cf86fd9962701396457e50ab0d6cc78aa29a5ebc/contracts/utils/math/SafeCast.sol

MINOR

GLOBAL-01 | THIRD-PARTY DEPENDENCY USAGE

Issue	Severity	Location	Status
Design Issue	Minor		Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treatsthird party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can becompromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severeimpacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
ICindexSwap public router;
address constant DAI = 0x6B175474E89094C44Da98b954EedeAC495271d0F;
address constant STETH = 0xae7ab96520DE3A18E5e111B5EaAb095312D7fE84;
address constant sDAI = 0x83F20F44975D03b1b09e64809B757c47f942BEeA;
mapping (address => AggregatorV3Interface) public oracles;
address[] public supportAssets;
```

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team toconstantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

MINOR

VCC-04| MISSING ZERO ADDRESS VALIDATION

Issue	Severity	Location	Status
Volatile Code	Minor	Vault.sol (v1): 81	Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leadingto unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent lossof those tokens.

```
81 PROTOCOL_FEE_RESERVE = _protocolFeeReserve;
```

- _protocolFeeReserve is not zero-checked before being used

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

INFORMATIONAL

VCC-05 | POTENTIAL DIVIDE BY ZERO

Issue	Severity	Location	Status
Logical Issue	Informational	Vault.sol (v1): 139, 255	Resolved

Description

```
139 uint256 amount1 = amount0 * (prices[0] / (10 ** decimals[0])) * weights
[1] / weights[0] / (prices[1] / (10 ** decimals[1]));
```

The expression amount0 * (prices[0] / (10 ** decimals[0])) * weights[1] / weights[0] / (prices[1] / (10 **decimals[1])) may divide by zero.

```
255 return totalSupply() > 0 ? totalSupply() * PRECISION / value :
PRECISION;
```

The expression totalSupply() * PRECISION / value may divide by zero.

Recommendation

It is recommended to either reformulate the divisor expression, or to use conditionals or require statements to rule out thepossibility of a divide-by-zero.

INFORMATIONAL

VCC-06 | POTENTIAL REVERSION IN PROTOCOL FEE CALCULATION

Issue	Severity	Location	Status
Logical Issue	Informational	Vault.sol (v1): 224, 225	Resolved

Description

The calProtocolFee() function in the contract is designed to calculate protocol fees based on the interest earned from the SavingsDaiMarket and StEthMarket . However, there is a possibility of reversion due to the way these fees arecalculated. Specifically, the function may revert if (sDaiAmount * exchangeRate) < assetAmounts[SavingsDaiMarket.sDAI] or sETHAmount < assetAmounts[StEthMarket.stETH] . The assetAmounts for both SavingsDaiMarket.sDAI and StEthMarket.stETH are updated in the updateAssetAmounts() function, where theyare set as sDaiAmount * exchangeRate and sETHAmount , respectively. Therefore, any decrease in the sDaiAmount or sETHAmount could potentially lead to a reversion when calculating the protocol fees.

Description

```
function updateAssetAmounts() internal {
       uint256 sDaiAmount = SavingsDaiMarket.balanceOf(address(this));
       uint256 exchangeRate = SavingsDaiMarket.exchangeRate();
       uint256 sETHAmount = StEthMarket.balanceOf(address(this));
       assetAmounts[SavingsDaiMarket.sDAI] = sDaiAmount * exchangeRate;
       assetAmounts[STETH] = sETHAmount;
}

/*
  *@dev Calculate fees
  */
function calProtocolFee() internal {
       if (totalSupply() > 0) {
       //sDAI amount
       uint256 sDaiAmount = SavingsDaiMarket.balanceOf(address(this));
       uint256 exchangeRate = SavingsDaiMarket.exchangeRate();
       //stETH amount
       uint256 sETHAmount = StEthMarket.balanceOf(address(this));
       //Interest-earning assets during this period
       uint256 sDaiInterestAmount = (sDaiAmount * exchangeRate) - assetAmounts[SavingsDaiMarket.sDAI];
       uint256 sETHInterestAmount = sETHAmount - assetAmounts[StEthMarket.stETH];
       //Calculate protocol fees
       uint256 sDaiFeeAmount = sDaiInterestAmount * protocolFee / 100 /1e18;
       uint256 sETHFeeAmount = sETHInterestAmount * protocolFee / 100;
       if (sDaiFeeAmount > 0) {
         TransferHelper.safeTransfer(SavingsDaiMarket.sDAI,PROTOCOL_FEE_RESERVE, sDaiFeeAmount);
       }
       if (sETHFeeAmount > 0) {
          TransferHelper.safeTransfer(STETH, PROTOCOL_FEE_RESERVE,sETHFeeAmount);
}
       emit ProtocolFee(sDaiFeeAmount, sETHFeeAmount);
     }
}
```

Recommendation

Since the values are determined by the third-party contracts sDAI and sETHAmount that are not in the audit scope, weassume the correctness. We would like the client to confirm the correctness of the current code. Consider handling the case when (sDaiAmount * exchangeRate) < assetAmounts[SavingsDaiMarket.sDAI] and sETHAmount <assetAmounts[StEthMarket.stETH] .

Disclaimer

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum	Cosmos
Eos	Substrate

TECH STACK

Python	Solidity
Rust	c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

September 5th 2024