MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

SECURITY ASSESSMENT

2021

MDEX TOKEN AUDIT REPORT

15 NOV 2021

SECURITY ASSESMENT

TABLE OF CONTENTS

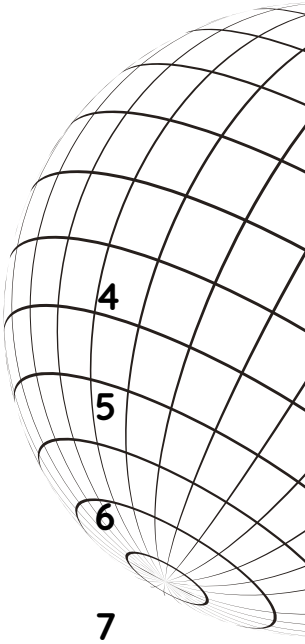
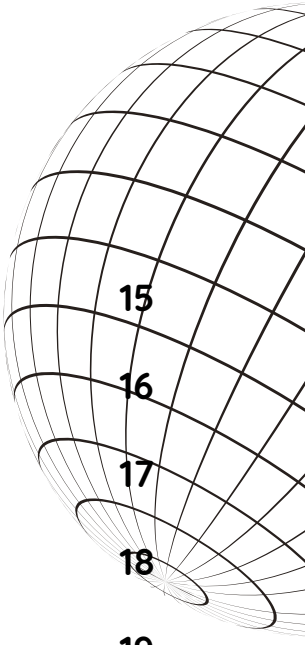


TABLE OF CONTENTS



SECURITY ASSESMENT

SUMMARY

This report has been prepared for MDEX smart contracts, one the leading currency trading platforms on the heco chain, to discover issues and vulnerabilities in the source code as well as any dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing static analysis and manual review techniques.

The auditing process pays special attention to the following considerations:

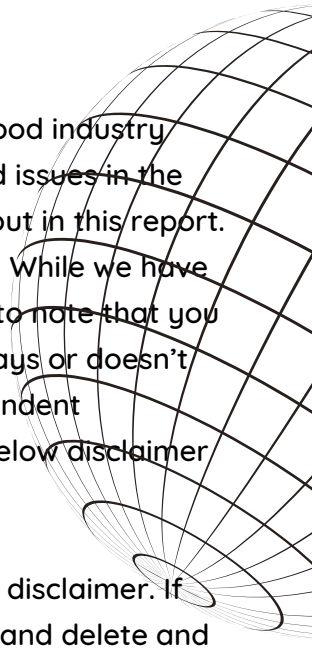
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by security experts.

The security assessment resulted in 13 findings that ranged from minor to informational. We recommend addressing these findings as potential improvements that can benefit the long run. We have done rounds of communications over the general understanding and the MDEX team has resolved the questions promptly.

Overall the source code is well written with security practices. The business logic is comprehensive and implemented accordingly. Yet we suggest a few recommendations that could better serve the project from the security perspective:

1. Enhance general coding practices for better structures of source codes;
2. Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
3. Provide more comments per each function for readability, especially contracts that are verified in public.
4. Provide more transparency on privileged activities once the protocol is live and gradually give the privileged permissions to the community once the protocol reaches a certain level of decentralization.

DISCLAIMER



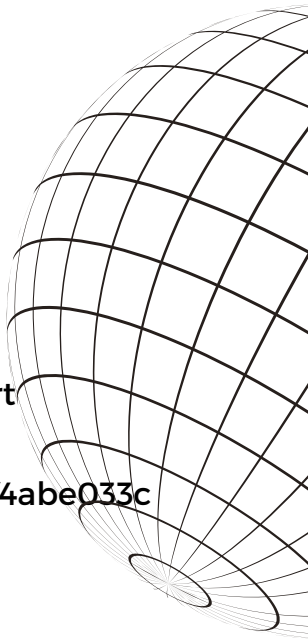
This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In 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 below disclaimer below – please make sure to read it in full.

DISCLAIMER: 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 MotechAudit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (MotechAudit) owe no duty of care towards you or any other person, nor does MotechAudit 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 MotechAudit 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, MotechAudit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against MotechAudit, 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 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.

SECURITY ASSESMENT

BACKGROUND



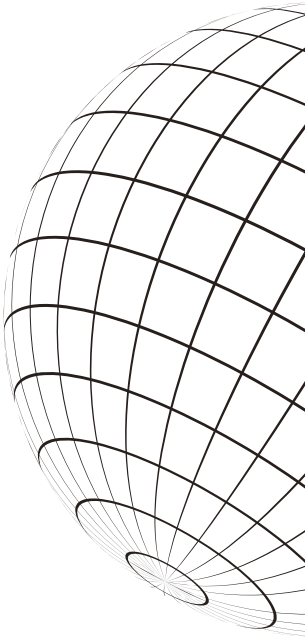
MotechAudit was commissioned by MDEX to perform an audit of smart contracts:
https://hecoinfo.com/address/0x25d2e80cb6b86881fd7e07dd263fb79f4abe033c
The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

SECURITY ASSESMENT

AUDIT DETAILS



AUDITED PROJECT

MDEX

DEPLOYER ADDRESS

0x06f46644D6e6d044aB008fb23bDC5Bf3529Bf3f0

CLIENT CONTACTS:

MDEX team

BLOCKCHAIN

Heco Chain Project

WEBSITE:

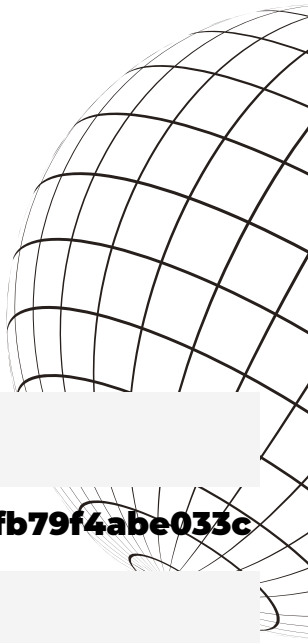
https://mdex.com/

MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

SECURITY ASSESMENT

CONTRACT DETAILS



Token contract details for Oct-11-2021

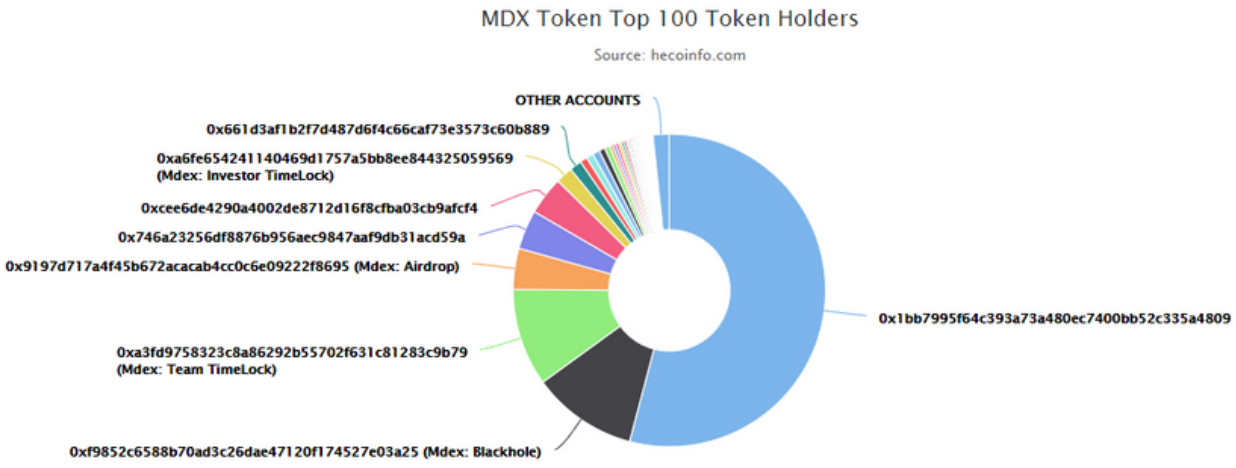
Contract name	**MDEX**
Contract address	**0x25d2e80cb6b86881fd7e07dd263fb79f4abe033c**
Total supply	**661,276,235.668807 MDX**
Token ticker	**MDX Token (MDX)**
Decimals	**18**
Token holders	**248,412**
Transactions count	**136,046,350**
Top 100 holders dominance	**97.6553%**
Contract deployer address	**0x06f46644D6e6d044aB008fb23bDC5Bf3529Bf3f0**
Contract's current owner address	**0x06f46644D6e6d044aB008fb23bDC5Bf3529Bf3f0**

SECURITY ASSESMENT

MDEX TOKEN DISTRIBUTION

The top 100 holders collectively own 98.32% (650,189,066.52 Tokens) of MDX Token

Token Total Supply: 661,283,036.71 Token | Total Token Holders: 248,408

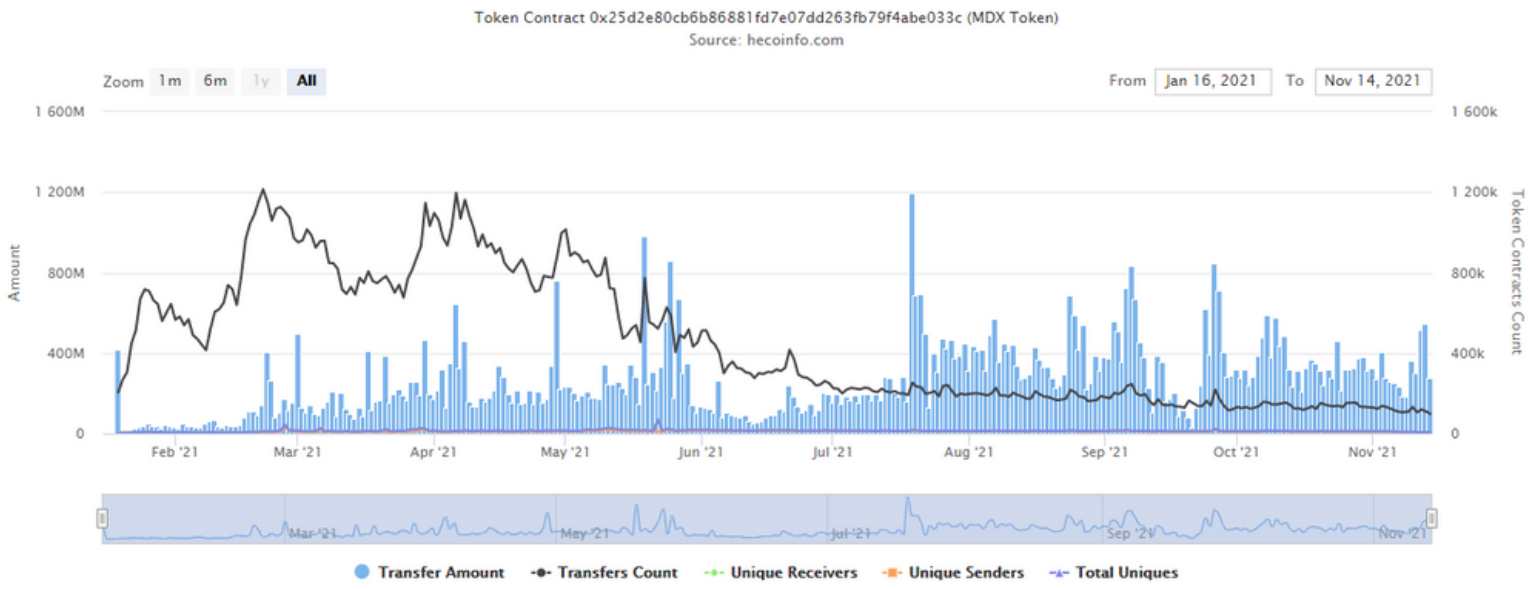


(A total of 650,189,066.52 tokens held by the top 100 accounts from the total supply of 661,283,036.71 token)

MDEX TOKEN CONTRACT INTERACTION DETAILS

Time Series: Token Contract Overview

Tue 19, Jan 2021 - Sun 14, Nov 2021



MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

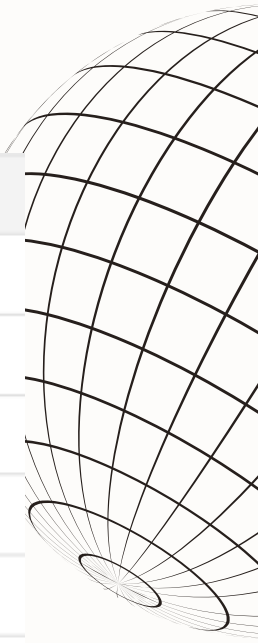
SECURITY ASSESMENT

TOP 10 TOKEN HOLDERS

Rank	Address	Quantity	Percentage	Value	Analytics
1	0x1bb7995f64c393a73a480ec7400bb52c335a4809	357,864,312.770878	54.1172%	$316,260,797.09	
2	Mdex: Blackhole	72,105,057.7317765	10.9039%	$63,722,484.25	
3	Mdex: Team TimeLock	66,666,672	10.0815%	$58,916,338.05	
4	Mdex: Airdrop	28,105,812.593686	4.2502%	$24,838,371.35	
5	0x746a23256df8876b956aec9847aaf9db31acd59a	26,881,388.0568819	4.0651%	$23,756,292.29	
6	0xcee6de4290a4002de8712d16f8cfba03cb9afcf4	26,037,944.7924539	3.9375%	$23,010,903.52	
7	Mdex: Investor TimeLock	11,666,670	1.7643%	$10,310,361.28	
8	0x661d3af1b2f7d487d6f4c66caf73e3573c60b889	8,339,694.79237352	1.2612%	$7,370,163.57	
9	Mdex: Brand TimeLock	5,000,000	0.7561%	$4,418,725.00	
10	0x020c15e7d08a8ec7d35bcf3ac3ccbf0bbf2704e6	4,841,337.41073157	0.7321%	$4,278,507.73	

source:https://hecoinfo.com/

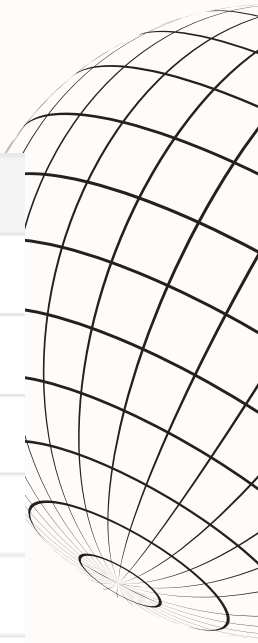
FINDINGS



ID	Severity	Response
CTK-MDEX#2-01	minor	Pending
CTK-MDEX#2-02	minor	Resolved
CTK-MDEX#2-03	minor	Pending
CTK-MDEX-01	minor	Pending
CTK-MDEX-02	minor	Pending
CTK-MDEX-03	minor	Pending
CTK-MDEX-04	minor	Pending
CTK-MDEX-05	minor	Pending
CTK-MDEX-06	minor	Pending
CTK-MDEX-07	minor	Pending
CTK-MDEX-08	Informational	Pending
CTK-MDEX-09	Informational	Pending
CTK-MDEX-10	informational	Pending
CTK-MDEX-11	minor	Pending
CTK-MDEX-12	minor	Pending

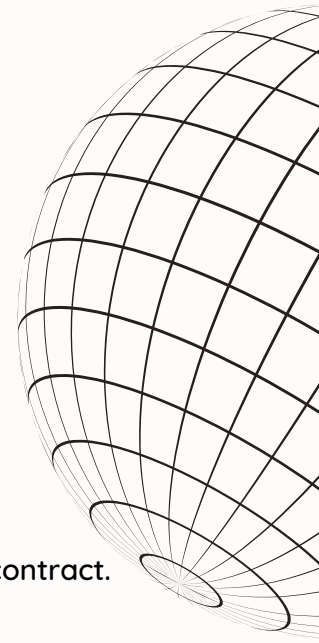
SECURITY ASSESMENT

CTK-MDEX#2-01 | UNDECLARED VARIABLE



ID	Severity	Response
CTK-MDEX#2-01	minor	Pending
CTK-MDEX#2-02	minor	Resolved
CTK-MDEX#2-03	minor	Pending
CTK-MDEX-01	minor	Pending
CTK-MDEX-02	minor	Pending
CTK-MDEX-03	minor	Pending
CTK-MDEX-04	minor	Pending
CTK-MDEX-05	minor	Pending
CTK-MDEX-06	minor	Pending
CTK-MDEX-07	minor	Pending
CTK-MDEX-08	Informational	Pending
CTK-MDEX-09	Informational	Pending
CTK-MDEX-10	informational	Pending
CTK-MDEX-11	minor	Pending
CTK-MDEX-12	minor	Pending

SECURITY ASSESMENT
CTK-MDEX#2-01 | UNDECLARED VARIABLE



Type	Severity	Location
Compiler Error	Minor	bscContracts/oracle.sol: consult()

Description

There is a letter q at the end of line 273, this causes an error when compiling the contract.

Recommendation

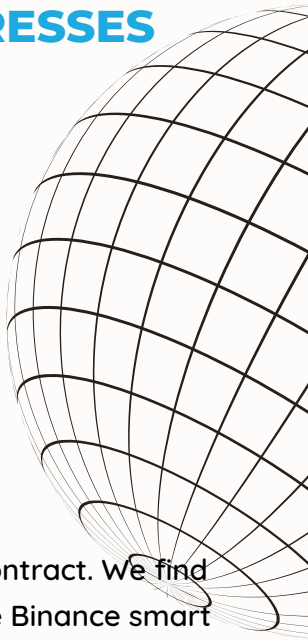
We assume this is a typo, the letter q should be removed.

Alleviation

The update has been applied here:
https://github.com/mdexSwap/contracts/commit/c9de3a25d4db6dc3e0c5231f4428b46232 e104f1

SECURITY ASSESMENT

CTK-MDEX#2-03 | INCORRECT CONTRACT ADDRESSES



Type	Severity	Location
Volatile Code	Minor	bscContracts/repurchase.sol: L850-L853

Description

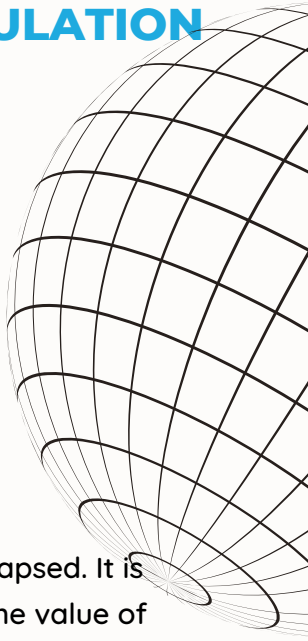
Multiple contracts' address is defined from line 850 to line 853 in the Repurchase contract. We find that those addresses belong to contracts deployed in the Heco chain instead of the Binance smart chain.

Recommendation

Update addresses to point them to specific contracts deployed in the Binance smart chain.

SECURITY ASSESMENT

CTK-MDEX-01 | DANGEROUS TIME-BASED CALCULATION



Type	Severity	Location
Volatile Code	Minor	oracle/Oracle.sol: consult()

Description

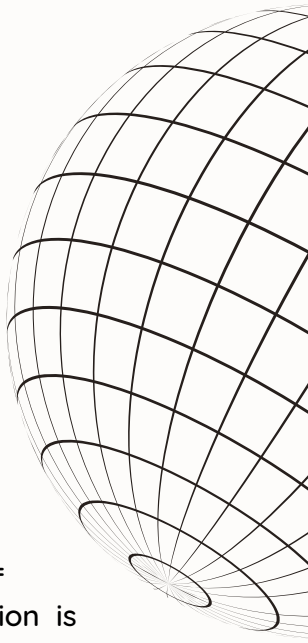
In function consult() , priceAverage would be updated based on the variable timeElapsed. It is possible to call update() first and then immediately call consult() .In this scenario, the value of timeElapsed would be rather small, and thus priceAverage could get a relatively large value.

Recommendation

Recommend adding a require() check to make sure a minimal period of time between the calls of update and consult(). Also, it is not recommended updating the value of priceAverage in consult() .

SECURITY ASSESMENT

CTK-MDEX-02 | UNREACHABLE FUNCTION



Type	Severity	Location
Volatile Code	Minor	governance/Timelock.sol: setPendingAdmin()

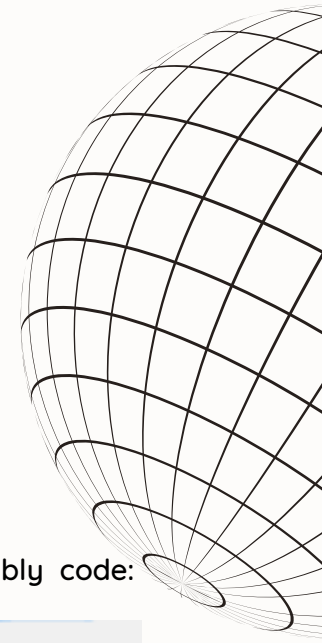
Description

Function setPendingAdmin has a require statement checking msg.sender == address(this), which means external calls are forbidden. However, the function is never called in the codebase.

Recommendation

Recommend either modify the require() check or add a helper function in the contract to call the setter function.

SECURITY ASSESMENT
CTK-MDEX-03 | WRONG ASSEMBLY CODE



Type	Severity	Location
Compiler Error	Minor	heco/Factory.sol: constructor() of MdexERC20

Description

The local variable chainId is assigned to chainid, which is not a valid assembly code:

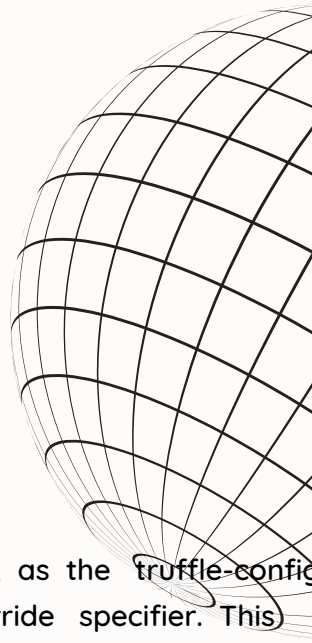
```
constructor() public {
    uint chainId;
    assembly {
        chainId := chainid
    }
    ...
}
```

Recommendation

Recommend correcting the assembly code to chainId := chainid() .

SECURITY ASSESMENT

CTK-MDEX-04 | MISSING OVERRIDE SPECIFIER



Type	Severity	Location
Compiler Error	Minor	heco/Factory.sol

Description

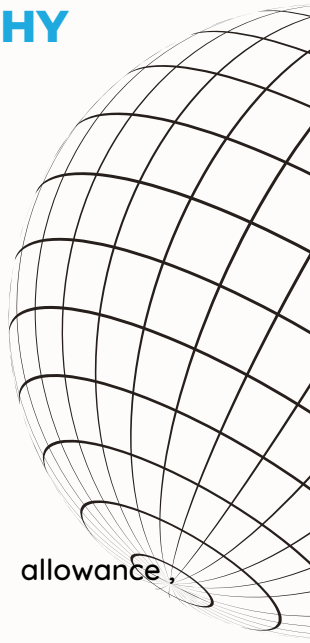
Based on the assumption that the contracts are to be deployed using 0.6.12, as the truffle-config file is using 0.6.12. There are multiple showings of functions lacking the override specifier. This will lead to failure of compiling the contracts with solidity 0.6.12.

Recommendation

For the inheriting functions, it is required to add virtual to every non-interface function intended to override, and to add override to the overriding functions, according to the Solidity 0.6.0 Breaking Changes .

SECURITY ASSESMENT

CTK-MDEX-05 | WRONG INHERITANCE HIERARCHY



Type	Severity	Location
Compiler Error	Minor	heco/Factory.sol: contract MdexPair

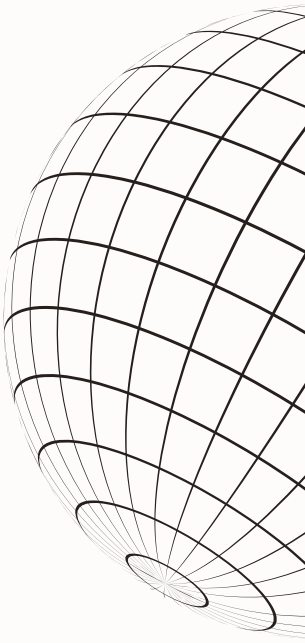
Description

1. There are compiler errors for DOMAIN_SEPARATOR , PERMIT_TYPEHASH , allowance , balanceOf , decimals , name , nonces , symbol , and totalSupply :
   Derived contract must override function "xxx". Two or more base classes define function with same name and parameter types. Since one of the bases defines a public state variable which cannot be overridden, you have to change the inheritance layout or the names of the functions.

2. There are compiler errors for approve , permit , transfer and transferFrom :
   Derived contract must override function "xxx". Two or more base classes define function with same name and parameter types.

SECURITY ASSESMENT

CTK-MDEX-06 | WRONG CONSTANT VALUE



Type	Severity	Location
Compiler Error	Minor	mainnet/CoinChef.sol

Description

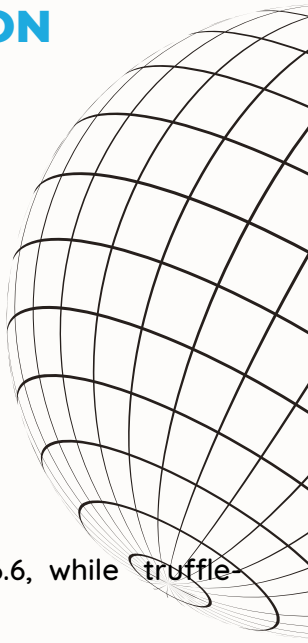
constant mdxPerBlock is assigned to 100 ** 1e18 , which seems too large.

Recommendation

It seems the value should be 100 * 1e18 instead of 100 ** 1e18 .

SECURITY ASSESMENT

CTK-MDEX-07 | INCONSISTENT SOLIDITY VERSION



Type	Severity	Location
Compiler Error	Minor	heco/Router.sol oracle/Oracle.sol

Description

The contract versions of heco/Router.sol and oracle/Oracle are locked at 0.6.6, while truffle-config is using 0.6.12.

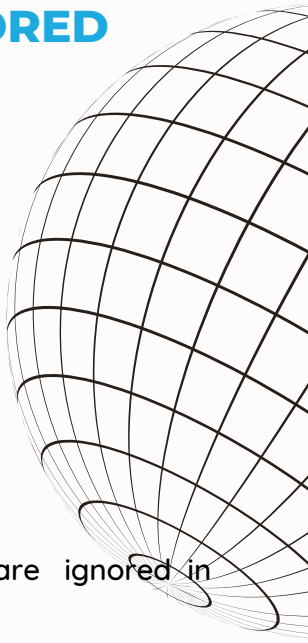
Recommendation

It is okay to try different compiler versions during the development stage.

However, we recommend locking the contract version when it reaches the production stage, and in this case seems 0.6.12 is more compatible.

SECURITY ASSESMENT

CTK-MDEX-08 | FUNCTION RETURN VALUE IGNORED



Type	Severity	Location
Control Flow	Informational	CoinChef.sol: L85, L393, L395

Description

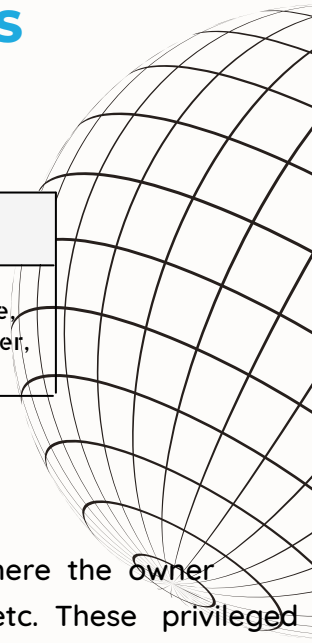
1. Return values of function IERC20(_addLP).approve(sushiChef,uint256(- 1)) are ignored in function addSushiLP(address) .
2. Return values of mdx.transfer(_to,mdxBal);mdx.transfer(_to,_amount) are ignored in the function safeMdxTransfer() .

Recommendation

IWe advise developers to handle the return values of aforementioned functions to check if the transfer is executed without any error.

SECURITY ASSESMENT

CTK-MDEX-09 | OVER PRIVILEGED OWNERSHIPS



Type	Severity	Location
Control Flow	Informational	Airdrop, AirdropMDX, Repurchase, HecoPool, MdxTokenHeco, Router, SwapMining, CoinChef

Description

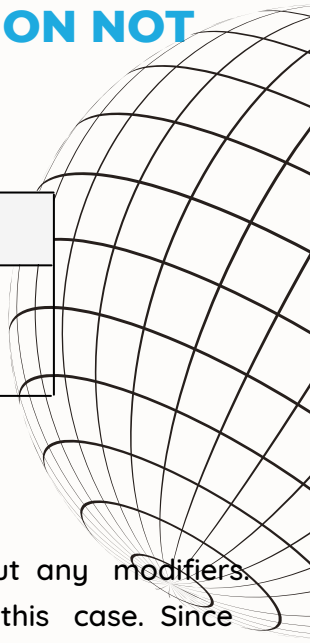
There are functions with modifier onlyOwner in the mentioned contracts, where the owner account could set some state variables, new airdrops, add new LP tokens, etc. These privileged behaviors could all potentially damage the economic system if abused without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing; or gradually migrate to a timelock plus multisig governing procedure and let the community to monitor in respect of transparency considerations

SECURITY ASSESMENT

CTK-MDEX-10 | `MINTER()` FUNCTION PERMISSION NOT RESTRICTED



Type	Severity	Location
Control Flow	Informational	MdxToken: setMinter()

Description

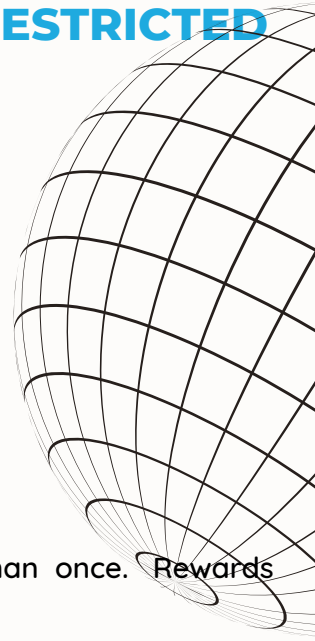
In MdxToken, function setMinter has its scope declared as external, without any modifiers. Everyone could call setMinter and let their own address be the minter in this case. Since function mint is only restricted by checking if minter is equal to msg.sender in onlyMinter, with the unprotected function setMinter, everyone could mint unlimitedly.

Recommendation

Consider assigning the contract creator as the first minter and then transfer to a different one i necessary. We understand that the team had handled this well during the post deployment stage as if such a thing happens the MDX could be simply replaced with a newly created contract.

SECURITY ASSESMENT

CTK-MDEX-11 | `ADD()` FUNCTION INPUT NOT RESTRICTED



Type	Severity	Location
Volatile Code	minor	CoinChef, Airdrop, AirdropMDX, HecoPool

Description

The comments mentioned // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.

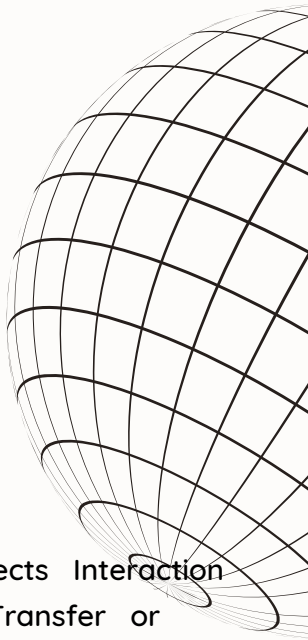
The total amount of reward in function updatePool() will be incorrectly calculated if the same LP token is added into the pool more than once in function add(). However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue. The current implementation is relying on the trust of the owner to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

Using mapping of addresses -> booleans, which can restrict the same address being added twice.

SECURITY ASSESMENT

CTK-MDEX-12 | CHECKS EFFECTS INTERACTION PATTERN NOT USED



Type	Severity	Location
Volatile Code	minor	CoinChef, Airdrop, AirdropMDX, HecoPool

Description

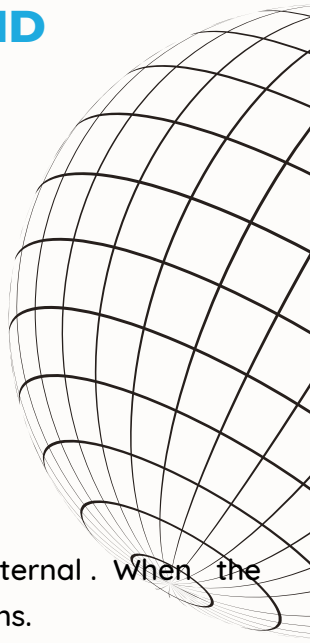
In functions deposit() and withdraw() of the four contracts, the Checks Effects Interaction Pattern is not strictly followed. Using interfaces, the implementation of safeTransfer or safeTransferFrom are unknown and may have a malicious logical implementation that calls back to the function deposit(). This is dangerous to the calculation like the user's balance, the pool's totalAmount, etc.

Recommendation

We advise developers to strictly follow the Checks-Effects-Interactions Pattern to avoid reentrancy and potential assets lost.

SECURITY ASSESMENT

CTK-MDEX-13 | PROPER USAGE OF `PUBLIC` AND `EXTERNAL`



Type	Severity	Location
Gas Optimization	Informational	General

Description

public functions that are never called by the contract could be declared external. When the inputs are arrays external functions are more efficient than public functions.

Recommendation

Consider using the external attribute for functions never called from the contract.

- newAirdrop(uint256,uint256,uint256): Airdrop.newAirdrop(uint256,uint256,uint256) (assets/Airdrop.sol#83-96)
- setCycle(uint256): Airdrop.setCycle(uint256) (assets/Airdrop.sol#106-108)
- add(uint256,IERC20,bool): Airdrop.add(uint256,IERC20,bool) (assets/Airdrop.sol#112-125)
- set(uint256,uint256,bool): Airdrop.set(uint256,uint256,bool) (assets/Airdrop.sol#128-134)
- deposit(uint256,uint256): Airdrop.deposit(uint256,uint256) (assets/Airdrop.sol#179-195)
- withdraw(uint256,uint256): Airdrop.withdraw(uint256,uint256) (assets/Airdrop.sol#198-213)
- emergencyWithdraw(uint256): Airdrop.emergencyWithdraw(uint256) (assets/Airdrop.sol#216-224)
- newAirdrop(uint256,uint256,uint256): AirdropMDX.newAirdrop(uint256,uint256,uint256) (assets/AirdropMDX.sol#79-92)
- setCycle(uint256): AirdropMDX.setCycle(uint256) (assets/AirdropMDX.sol#102-104)
- add(uint256,IERC20,bool): AirdropMDX.add(uint256,IERC20,bool) (assets/AirdropMDX.sol#108-122)
- set(uint256,uint256,bool): AirdropMDX.set(uint256,uint256,bool) (assets/AirdropMDX.sol#125-131)
- deposit(uint256,uint256): AirdropMDX.deposit(uint256,uint256) (assets/AirdropMDX.sol#186-205)
- withdraw(uint256,uint256): AirdropMDX.withdraw(uint256,uint256) (assets/AirdropMDX.sol#208-226)
- emergencyWithdraw(uint256): AirdropMDX.emergencyWithdraw(uint256) (assets/AirdropMDX.sol#229-240)
- setAmountIn(uint256): Repurchase.setAmountIn(uint256) (assets/Repurchase.sol#32-34)
- setEmergencyAddress(address): Repurchase.setEmergencyAddress(address) (assets/Repurchase.sol#36-39)

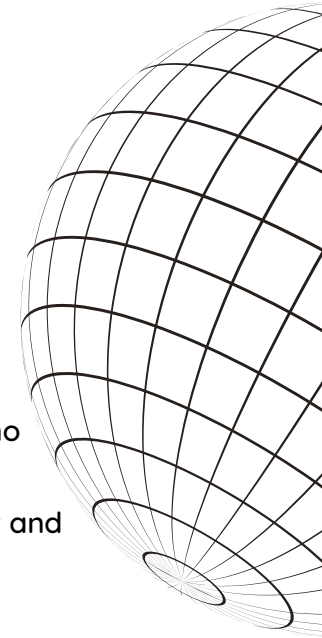
- addCaller(address): Repurchase.addCaller(address) (assets/Repurchase.sol#41-44)
- delCaller(address): Repurchase.delCaller(address) (assets/Repurchase.sol#46-49)
- getCaller(uint256): Repurchase.getCaller(uint256) (assets/Repurchase.sol#59-62)
- emergencyWithdraw(address): Repurchase.emergencyWithdraw(address) (assets/Repurchase.sol#78-81)
- propose(address[],uint256[],string[],bytes[],string): GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (governance/GovernorAlpha.sol#138-176)
- queue(uint256): GovernorAlpha.queue(uint256) (governance/GovernorAlpha.sol#178-187)
- execute(uint256): GovernorAlpha.execute(uint256) (governance/GovernorAlpha.sol#194-202)
- cancel(uint256): GovernorAlpha.cancel(uint256) (governance/GovernorAlpha.sol#204-217)
- getActions(uint256): GovernorAlpha.getActions(uint256) (governance/GovernorAlpha.sol#219-222)
- getReceipt(uint256,address): GovernorAlpha.getReceipt(uint256,address) (governance/GovernorAlpha.sol#224-226)
- castVote(uint256,bool): GovernorAlpha.castVote(uint256,bool) (governance/GovernorAlpha.sol#250-252)
- castVoteBySig(uint256,bool,uint8,bytes32,bytes32): GovernorAlpha.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (governance/GovernorAlpha.sol#254-261)
- __acceptAdmin(): GovernorAlpha.__acceptAdmin() (governance/GovernorAlpha.sol#283-286)
- __abdicate(): GovernorAlpha.__abdicate() (governance/GovernorAlpha.sol#288-291)
- __queueSetTimelockPendingAdmin(address,uint256): GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (governance/GovernorAlpha.sol#293-296)
- __executeSetTimelockPendingAdmin(address,uint256): GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (governance/GovernorAlpha.sol#298-301)
- mint(address,uint256): MdxToken.mint(address,uint256) (heco/MdxTokenHeco.sol#253-259)
- addMinter(address): MdxToken.addMinter(address) (heco/MdxTokenHeco.sol#261-264)
- delMinter(address): MdxToken.delMinter(address) (heco/MdxTokenHeco.sol#266-269)
- getMinter(uint256): MdxToken.getMinter(uint256) (heco/MdxTokenHeco.sol#279-282)
- setDelay(uint256): Timelock.setDelay(uint256) (governance/Timelock.sol#36-43)
- acceptAdmin(): Timelock.acceptAdmin() (governance/Timelock.sol#45-51)
- setPendingAdmin(address): Timelock.setPendingAdmin(address) (governance/Timelock.sol#53-58)
- queueTransaction(address,uint256,string,bytes,uint256): Timelock.queueTransaction(address,uint256,string,bytes,uint256) (governance/Timelock.sol#60-69)
- cancelTransaction(address,uint256,string,bytes,uint256): Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (governance/Timelock.sol#71-78)

- executeTransaction(address,uint256,string,bytes,uint256): Timelock.executeTransaction(address,uint256,string,bytes,uint256) (governance/Timelock.sol#80-105)
- price(address,uint256): MdexPair.price(address,uint256) (heco/Factory.sol#343-352)
- quote(uint256,uint256,uint256): MdexFactory.quote(uint256,uint256,uint256) (heco/Factory.sol#438-442)
- getAmountsOut(uint256,address[]): MdexFactory.getAmountsOut(uint256,address[]) (heco/Factory.sol#464-472)
- getAmountsIn(uint256,address[]): MdexFactory.getAmountsIn(uint256,address[]) (heco/Factory.sol#475-483)
- setHalvingPeriod(uint256): HecoPool.setHalvingPeriod(uint256) (heco/HecoPool.sol#84-86)
- setMdxPerBlock(uint256): HecoPool.setMdxPerBlock(uint256) (heco/HecoPool.sol#89-92)
- addMultLP(address): HecoPool.addMultLP(address) (heco/HecoPool.sol#98-102)
- getMultLPAddress(uint256): HecoPool.getMultLPAddress(uint256) (heco/HecoPool.sol#112-115)
- setPause(): HecoPool.setPause() (heco/HecoPool.sol#117-119)
- setMultLP(address,address): HecoPool.setMultLP(address,address) (heco/HecoPool.sol#121-125)
- replaceMultLP(address,address): HecoPool.replaceMultLP(address,address) (heco/HecoPool.sol#127-140)
- add(uint256,IERC20,bool): HecoPool.add(uint256,IERC20,bool) (heco/HecoPool.sol#144-160)
- set(uint256,uint256,bool): HecoPool.set(uint256,uint256,bool) (heco/HecoPool.sol#163-169)
- setPoolCorr(uint256,uint256): HecoPool.setPoolCorr(uint256,uint256) (heco/HecoPool.sol#172-175)
- deposit(uint256,uint256): HecoPool.deposit(uint256,uint256) (heco/HecoPool.sol#300-307)
- withdraw(uint256,uint256): HecoPool.withdraw(uint256,uint256) (heco/HecoPool.sol#367-374)
- emergencyWithdraw(uint256): HecoPool.emergencyWithdraw(uint256) (heco/HecoPool.sol#422-429)
- addPair(uint256,address,bool): SwapMining.addPair(uint256,address,bool) (heco/SwapMining.sol#85-101)
- setPair(uint256,uint256,bool): SwapMining.setPair(uint256,uint256,bool) (heco/SwapMining.sol#104-110)
- setMdxPerBlock(uint256): SwapMining.setMdxPerBlock(uint256) (heco/SwapMining.sol#113-116)
- addWhitelist(address): SwapMining.addWhitelist(address) (heco/SwapMining.sol#119-122)
- delWhitelist(address): SwapMining.delWhitelist(address) (heco/SwapMining.sol#124-127)
- setHalvingPeriod(uint256): SwapMining.setHalvingPeriod(uint256) (heco/SwapMining.sol#142-144)
- setRouter(address): SwapMining.setRouter(address) (heco/SwapMining.sol#146-149)
- setOracle(IOracle): SwapMining.setOracle(IOracle) (heco/SwapMining.sol#151-154)
- phase(): SwapMining.phase() (heco/SwapMining.sol#167-169)
- reward(): SwapMining.reward() (heco/SwapMining.sol#176-178)
- swap(address,address,address,uint256): SwapMining.swap(address,address,address,uint256) (heco/SwapMining.sol#226-260)
- takerWithdraw(): SwapMining.takerWithdraw() (heco/SwapMining.sol#263-284)
- getUserReward(uint256): SwapMining.getUserReward(uint256) (heco/SwapMining.sol#287-299)

- getPoolInfo(uint256): SwapMining.getPoolInfo(uint256) (heco/SwapMining.sol#302-313)
- addSushiLP(address): CoinChef.addSushiLP(address) (mainnet/CoinChef.sol#83-87)
- getSushiLPAddress(uint256): CoinChef.getSushiLPAddress(uint256) (mainnet/CoinChef.sol#97-100)
- add(uint256,IERC20,bool): CoinChef.add(uint256,IERC20,bool) (mainnet/CoinChef.sol#104-120)
- set(uint256,uint256,bool): CoinChef.set(uint256,uint256,bool) (mainnet/CoinChef.sol#123-129)
- setPoolCorr(uint256,uint256): CoinChef.setPoolCorr(uint256,uint256) (mainnet/CoinChef.sol#132-135)
- deposit(uint256,uint256): CoinChef.deposit(uint256,uint256) (mainnet/CoinChef.sol#232-259)
- withdraw(uint256,uint256): CoinChef.withdraw(uint256,uint256) (mainnet/CoinChef.sol#299-306)
- emergencyWithdraw(uint256): CoinChef.emergencyWithdraw(uint256) (mainnet/CoinChef.sol#354-361)
- mint(address,uint256): MdxToken.mint(address,uint256) (mainnet/MdxToken.sol#10-13)
- getBalance(): TeamTimeLock.getBalance() (timeLock/TeamTimeLock.sol#44-46)
- setBeneficiary(address): TeamTimeLock.setBeneficiary(address) (timeLock/TeamTimeLock.sol#71-74)

SECURITY ASSESMENT

CONCLUSION



Smart contracts contain owner privileges!

Motech Audit note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.