BLOCKHAT
SECURITY

MindX

Smart Contract Security Audit

Prepared by BlockHat

February 26th, 2024 - February 29th, 2024

BlockHat.io

contact@blockhat.io

Document Properties

Client	MindX
Version	0.1
Classification	Private

Scope

Repository	Commit Hash

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

MindX engaged BlockHat to conduct a security assessment on the MindX beginning on February 26th, 2024 and ending February 29th, 2024. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About MindX

Issuer	MindX
Website	
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

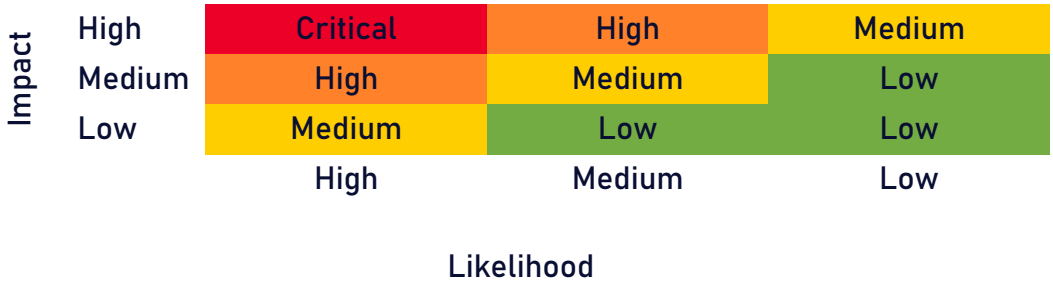
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the MindX implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 2 critical-severity, 1 high-severity, 3 medium-severity, 1 low-severity, 1 informational-severity vulnerabilities.

Vulnerabilities	Severity	Status
Unrestricted Access Control	CRITICAL	Not Fixed
Trading Enabled by Default	CRITICAL	Not Fixed
Lack of Fee Limits	HIGH	Not Fixed
Use of Outdated ERC20 and Ownable Contracts	MEDIUM	Not Fixed
Redundant Address Assignments	MEDIUM	Not Fixed
Misleading Function Names and Redundancies	MEDIUM	Not Fixed
Inefficient Use of Arithmetic Operations	LOW	Not Fixed
Unclear Purpose of Tier Timestamps	INFORMATIONAL	Not Fixed

3 Finding Details

A Mindx.sol

A.1 Unrestricted Access Control [CRITICAL]

Description:

Several functions that should be restricted to the owner are publicly accessible, posing a significant security risk.

Code:

Listing 1: Mindx.sol

```
901     function adding_isExcludedMaxTransactionAmount(address _a) public {
902         _isExcludedMaxTransactionAmount[_a] = true;
903         emit adding_isExcluded(_a);
904     }

906     function removing_isExcludedMaxTransactionAmount(address _a) public
            ↪ {
907         delete _isExcludedMaxTransactionAmount[_a];
908         emit removing_isExcluded(_a);
909     }

911     function adding_automatedMarketMakerPairs(address _a) public {
912         _automatedMarketMaker[_a] = true;
913         emit adding_automated(_a);
914     }

916     function removing_automatedMarketMakerPairs(address _a) public {
917         delete _automatedMarketMaker[_a];
918         emit removing_automated(_a);
919     }
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

Restrict access to sensitive functions by implementing appropriate access control checks.

Status - Not Fixed

A.2 Trading Enabled by Default [CRITICAL]

Description:

Trading is enabled by default in the constructor, which can lead to security risks and unintended trading before the contract setup is fully complete.

Code:

Listing 2: Mindx.sol

```
832         tradingEnabled = true;
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

Modify the contract to have trading disabled by default. Enable trading explicitly after all initializations are completed securely.

Status - Not Fixed

A.3 Lack of Fee Limits [HIGH]

Description:

There are no limits on the fees that can be set, potentially allowing for unreasonable or exploitative fee levels.

Code:

Listing 3: Mindx.sol

```
845     function taxChange(uint _b, uint _s) external onlyOwner {
846         liquidityFeeOnBuy = _b;
847         liquidityFeeOnSell = _s;
848
849         emit tax_change(_b, _s);
850     }
851
852     function divChange(uint _b, uint _s) external onlyOwner {
853         liquidityFeeOnBuy = _b;
854         liquidityFeeOnSell = _s;
855
856         emit tax_fee(_b, _s);
857     }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

Implement a maximum fee limit to protect users from excessive charges.

Status - Not Fixed

A.4 Use of Outdated ERC20 and Ownable Contracts [MEDIUM]

Description:

The contract uses outdated versions of the ERC20 and Ownable contracts, which may lack recent security improvements and optimizations.

Code:

Listing 4: Mindx.sol

```
783 contract Mindx is ERC20, Ownable {
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Upgrade to the latest versions of these contracts to incorporate the latest security fixes and improvements.

Status - Not Fixed

A.5 Redundant Address Assignments [MEDIUM]

Description:

The TechTeam, TreasuryRevenue, and TreasuryOwner are all set to the same address, causing unnecessary redundancy in address assignments and mapping settings.

Code:

Listing 5: Mindx.sol

```
796     address public TechTeam = 0x60FF0d52212B896438E2f6f35c5A75e0229539db
            ↪ ;
797     address public TreasuryRevenue = 0
            ↪ x60FF0d52212B896438E2f6f35c5A75e0229539db;
798     address public TreasuryOwner = 0
            ↪ x60FF0d52212B896438E2f6f35c5A75e0229539db;
```

Listing 6: Mindx.sol

```
821         _automatedMarketMaker[msg.sender] = true;
822         _automatedMarketMaker[TechTeam] = true;
823         _automatedMarketMaker[TreasuryRevenue] = true;
824         _automatedMarketMaker[TreasuryOwner] = true;
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

Ensure that different roles are assigned to distinct addresses to reduce confusion and increase contract clarity.

Status - Not Fixed

A.6 Misleading Function Names and Redundancies [MEDIUM]

Description:

The function enableTrading is misleadingly named as it can also disable trading. Additionally, there are redundant functions for fee setting.

Code:

Listing 7: Mindx.sol

```
839     function enableTrading(bool _status) external onlyOwner {
840         require(!tradingEnabled, "Trading already enabled.");
841         tradingEnabled = _status;
842         emit enable_trading(_status);
843     }
```

Listing 8: Mindx.sol

```
845     function taxChange(uint _b, uint _s) external onlyOwner {
846         liquidityFeeOnBuy = _b;
847         liquidityFeeOnSell = _s;

849         emit tax_change(_b, _s);
850     }

852     function divChange(uint _b, uint _s) external onlyOwner {
853         liquidityFeeOnBuy = _b;
854         liquidityFeeOnSell = _s;

856         emit tax_fee(_b, _s);
857     }
```

Risk Level:

Likelihood – 3

Impact - 3

Recommendation:

Rename functions for clarity and remove redundant functions to simplify contract logic.

Status - Not Fixed

A.7 Inefficient Use of Arithmetic Operations [LOW]

Description:

The contract performs multiplication before division in share calculations, leading to potential rounding errors and inefficiencies.

Code:

Listing 9: Mindx.sol

```
829         uint techTeam = (total_Supply / 100) * 5
```

Listing 10: Mindx.sol

```
887         Taxation = (amount / 100) * Taxation;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Follow best practices for arithmetic operations to minimize rounding errors and gas costs.

Status - Not Fixed

A.8 Unclear Purpose of Tier Timestamps [INFORMATIONAL]

Description:

The utility of _tierTimestamp mapping is unclear, raising questions about its purpose and implementation.

Code:

Listing 11: Mindx.sol

```
895         _tierTimestamp[to] = block.timestamp;
896         _tierTimestamp[from] = block.timestamp;
```

Listing 12: Mindx.sol

```
921     function getTier(address account) public view returns (uint ) {
922         return _tierTimestamp[account];
923     }
```

Risk Level:

Likelihood – 1
Impact - 1

Recommendation:

Clarify the purpose of tier timestamps and ensure they are implemented securely and effectively.

Status - Not Fixed

4 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
INFO:Detectors:
Mindx.OwnerShare (token.sol#792) is never initialized. It is used in:
        - Mindx._transfer(address,address,uint256) (token.sol#866-899)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables
INFO:Detectors:
Mindx.constructor() (token.sol#820-835) performs a multiplication on the
    ↪  result of a division:
        - techTeam = (total_Supply / 100) * 5 (token.sol#829)
Mindx._transfer(address,address,uint256) (token.sol#866-899) performs a
    ↪ multiplication on the result of a division:
        - Taxation = (amount / 100) * Taxation (token.sol#887)
Mindx._transfer(address,address,uint256) (token.sol#866-899) performs a
    ↪ multiplication on the result of a division:
        - _owner_share = (Taxation / 100) * OwnerShare (token.sol#890)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply
INFO:Detectors:
Contract locking ether found:
        Contract Mindx (token.sol#783-924) has payable functions:
         - Mindx.receive() (token.sol#837)
```

```
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether
INFO:Detectors:
Mindx.divAdress(address,address)._tr (token.sol#859) lacks a zero-check
    ↪ on :
                - TreasuryRevenue = _tr (token.sol#860)
Mindx.divAdress(address,address)._to (token.sol#859) lacks a zero-check
    ↪ on :
                - TreasuryOwner = _to (token.sol#861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation
INFO:Detectors:
Address._revert(bytes,string) (token.sol#528-543) uses assembly
        - INLINE ASM (token.sol#536-539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage
INFO:Detectors:
Address._revert(bytes,string) (token.sol#528-543) is never used and
    ↪ should be removed
Address.functionCall(address,bytes) (token.sol#387-398) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (token.sol#400-406) is never
    ↪ used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (token.sol#408-420)
    ↪  is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (token.sol
    ↪ #422-442) is never used and should be removed
Address.functionDelegateCall(address,bytes) (token.sol#471-481) is never
    ↪  used and should be removed
Address.functionDelegateCall(address,bytes,string) (token.sol#483-496)
    ↪ is never used and should be removed
Address.functionStaticCall(address,bytes) (token.sol#444-454) is never
    ↪ used and should be removed
```

```
Address.functionStaticCall(address,bytes,string) (token.sol#456-469) is
    ↪ never used and should be removed
Address.isContract(address) (token.sol#370-372) is never used and should
    ↪  be removed
Address.sendValue(address,uint256) (token.sol#374-385) is never used and
    ↪  should be removed
Address.verifyCallResult(bool,bytes,string) (token.sol#516-526) is never
    ↪  used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (token.sol
    ↪ #498-514) is never used and should be removed
Context._msgData() (token.sol#551-554) is never used and should be
    ↪ removed
ERC20._burn(address,uint256) (token.sol#741-756) is never used and
    ↪ should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code
INFO:Detectors:
Pragma version^0.8.0 (token.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (token.sol#374-385)
    ↪ :
        - (success) = recipient.call{value: amount}() (token.sol#383)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (token.sol#422-442):
        - (success,returndata) = target.call{value: value}(data) (token.
            ↪ sol#432-434)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ token.sol#456-469):
        - (success,returndata) = target.staticcall(data) (token.sol#461)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ token.sol#483-496):
```

```
        - (success,returndata) = target.delegatecall(data) (token.sol
          ↪ #488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (token.sol#64) is not in
   ↪ mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (token.sol#66) is not in
   ↪ mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (token.sol#97) is not in
   ↪ mixedCase
Function IUniswapV2Router01.WETH() (token.sol#137) is not in mixedCase
Event Mindx.adding_isExcluded(address) (token.sol#811) is not in
   ↪ CapWords
Event Mindx.removing_isExcluded(address) (token.sol#812) is not in
   ↪ CapWords
Event Mindx.adding_automated(address) (token.sol#813) is not in CapWords
Event Mindx.removing_automated(address) (token.sol#814) is not in
   ↪ CapWords
Event Mindx.enable_trading(bool) (token.sol#815) is not in CapWords
Event Mindx.tax_change(uint256,uint256) (token.sol#816) is not in
   ↪ CapWords
Event Mindx.tax_Treasury(address,address) (token.sol#817) is not in
   ↪ CapWords
Event Mindx.tax_fee(uint256,uint256) (token.sol#818) is not in CapWords
Parameter Mindx.enableTrading(bool)._status (token.sol#839) is not in
   ↪ mixedCase
Parameter Mindx.taxChange(uint256,uint256)._b (token.sol#845) is not in
   ↪ mixedCase
Parameter Mindx.taxChange(uint256,uint256)._s (token.sol#845) is not in
   ↪ mixedCase
Parameter Mindx.divChange(uint256,uint256)._b (token.sol#852) is not in
   ↪ mixedCase
```

```
Parameter Mindx.divChange(uint256,uint256)._s (token.sol#852) is not in
    ↪ mixedCase
Parameter Mindx.divAdress(address,address)._tr (token.sol#859) is not in
    ↪  mixedCase
Parameter Mindx.divAdress(address,address)._to (token.sol#859) is not in
    ↪  mixedCase
Function Mindx.adding_isExcludedMaxTransactionAmount(address) (token.sol
    ↪ #901-904) is not in mixedCase
Parameter Mindx.adding_isExcludedMaxTransactionAmount(address)._a (token
    ↪ .sol#901) is not in mixedCase
Function Mindx.removing_isExcludedMaxTransactionAmount(address) (token.
    ↪ sol#906-909) is not in mixedCase
Parameter Mindx.removing_isExcludedMaxTransactionAmount(address)._a (
    ↪ token.sol#906) is not in mixedCase
Function Mindx.adding_automatedMarketMakerPairs(address) (token.sol
    ↪ #911-914) is not in mixedCase
Parameter Mindx.adding_automatedMarketMakerPairs(address)._a (token.sol
    ↪ #911) is not in mixedCase
Function Mindx.removing_automatedMarketMakerPairs(address) (token.sol
    ↪ #916-919) is not in mixedCase
Parameter Mindx.removing_automatedMarketMakerPairs(address)._a (token.
    ↪ sol#916) is not in mixedCase
Variable Mindx.RevenueShare (token.sol#791) is not in mixedCase
Variable Mindx.OwnerShare (token.sol#792) is not in mixedCase
Variable Mindx.TechTeam (token.sol#796) is not in mixedCase
Variable Mindx.TreasuryRevenue (token.sol#797) is not in mixedCase
Variable Mindx.TreasuryOwner (token.sol#798) is not in mixedCase
Variable Mindx._tierTimestamp (token.sol#800) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (token.sol#552)" inContext (token.sol
    ↪ #546-555)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256
    ↪ ,uint256,uint256,address,uint256).amountADesired (token.sol#142)
    ↪ is too similar to IUniswapV2Router01.addLiquidity(address,address
    ↪ ,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
    ↪ (token.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-too-similar
INFO:Detectors:
Mindx.swapping (token.sol#793) is never used in Mindx (token.sol
    ↪ #783-924)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable
INFO:Detectors:
Mindx.OwnerShare (token.sol#792) should be constant
Mindx.RevenueShare (token.sol#791) should be constant
Mindx.TechTeam (token.sol#796) should be constant
Mindx.swapping (token.sol#793) should be constant
Mindx.uniswapV2Pair (token.sol#795) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

5 Conclusion

We examined the design and implementation of MindX in this audit and found several issues of various severities. We advise MindX team to implement the recommendations contained in all 8 of our findings to further enhance the code's security. It is of utmost priority to start by addressing the most severe exploit discovered by the auditors then followed by the remaining exploits, and finally we will be conducting a re-audit following the implementation of the remediation plan contained in this report.

We would much appreciate any constructive feedback or suggestions regarding our methodology, audit findings, or potential scope gaps in this report.

BLOCKHAT
SECURITY

For a Smart Contract Audit, contact us at contact@blockhat.io