Guard

Smart contracts security assessment

Final report

Tariff: Standard

MtopSwap NFT

August 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

Standard ERC721 token contract.

This contract is deployed in the Harmony network at one1zpfm6tetx0q6etvj0wclej00nh6qfd90arzmzh, the code is available at this link in the official explorer.

ERC721 interface is realized with the use of OpenZeppelin libraries, which is considered the best practice.

Oracle contracts were out of the scope of this audit, their addresses: ONE_ORACLE - 0x12f9c4b725457c48a3aD761530D8a7e5282E57E7, MTOP_ORACLE - 0xDeEB347937E280F459Ea6999b5a5C17684c16096.

Name	MtopSwap NFT
Audit date	2022-08-05 - 2022-08-09
Language	Solidity
Platform	Harmony

Contracts checked

Name	Address
MtopNFT	https://explorer.harmony.one/address/0x1053bd2f2b33c1acad927bb1fcc9ef9df404b4af

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	not 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Outdated Compiler Version	passed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

No issues were found

Medium severity issues

1. Blocking the buy() function (MtopNFT)

If the owner changes the addresses of the `feeCollector`, `ONE_ORACLE`, and `MTOP_ORACLE` variables using the `setFeeCollector()`, `setOneOracle()`, or `setMtopOracle()` functions, he can block calls to the `buy()` function.

For example, if `feeCollector` is a contract without `recieve()` and `fallback()` functions, then an exception will be thrown at 116L.

```
function buy(address to, Tier tier) public payable {
    ...
    (uint256 _mtopAmount, uint256 _oneAmount, ) = getTierCost(tier); // oracles
exceptions
    ...
    payable(feeCollector).transfer(_oneAmount); // feeCollector exception
    ...
    }
```

Team response: The “Owner Blocking the buy() function” finding allows the owner of the contract to block subscription purchases, i.e. minting new Subscription NFTs. This doesn't affect the security of the contract, or the owners of the NFTs themselves. Additionally, the contract doesn’t hold any tokens.

Low severity issues

1. Gas optimization (MtopNFT)

Visibility of the functions `batchGift()`, `buy()`, `supportsInterface()`, `baseURI()`, `tokenURI()` can be declared as `external` to save gas.

2. Few events (MtopNFT)

Many functions from the contract lack events:

1. `setFeeCollector()`
2. `setOneOracle()`
3. `setMtopOracle()`
4. `gift()`
5. `buy()`

Team response: Noted. The impact is that we can't do event based data analysis on this contract for these functions.

3. Functions lacks validation of input parameters (MtopNFT)

The contract functions does not check the input addresses against a null address:

1) `_newFeeCollector` in `setFeeCollector()`

2) `_newOneOracle` in `setOneOracle()`

3) `_newMtopOracle` in `setMtopOracle()`

Team response: Noted. Generally this is not desirable. We ensure and commit that the admin won't set a zero address for the mentioned functions. Should be noted that this poses no security issues with the user's Token.

4. Variables visibility modifier (MtopNFT)

State variables `feeCollector`, `SECONDS_PER_DAY`, `MTOP`, `MTOP_DECIMALS`, `MTOP_ORACLE`, `ONE_ORACLE` do not have the specified visibility modifier.

5. Typos (MtopNFT)

Typos reduce the code's readability.

118L 'valueable' should be replaced with 'valuable'

6. Price calculation through unknown oracles (MtopNFT)

The calculation of the price in this contract occurs with the help of unknown oracles, this can lead to unforeseen consequences for the user or the owner of the contract.

Recommendation: It is recommended to specify addresses of oracles in external project sources or change the visibility modifiers for oracle variables in the contract. Users are encouraged to conduct their research on the oracles of this contract.

Team response: The addresses of our Oracles can be found to the following external project sources:

- Website
- Whitepaper
- Talk Forum Grant Proposal Thread
- Harmony Explorer 1
- Harmony Explorer 2

7. Constructor lacks validation of input parameters (MtopNFT)

The contract constructor does not check the addresses `_mtop` , `_oneOracle` , `_mtopOracle`and `_feeCollector` against a null address.

Team response: Noted. Generally this is not desirable. We ensure and commit that the admin won't set a zero address for the mentioned functions. Should be noted that this poses no security issues

with the user’s Token.

8. Floating Pragma (MtopNFT)

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Team response: The pragma mismatch happens because there is a slight mismatch in versions between the NFT Contracts & the Oracles. The NFT Contract is version 0.8.4 & the oracle is 0.8.9. However, the Oracles & the NFT contract use the same compiler generation i.e. 0.8. We have tested the contracts extensively and haven't identified any problems.

Conclusion

MtopSwap NFT MtopNFT contract was audited. 1 medium, 8 low severity issues were found.

Oracle contracts were out of the scope of this audit, their addresses: ONE_ORACLE - 0x12f9c4b725457c48a3aD761530D8a7e5282E57E7, MTOP_ORACLE - 0xDeEB347937E280F459Ea6999b5a5C17684c16096.

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.

Slither output

```
MtopNFT.buy(address,MtopNFT.Tier) (contracts/MtopNFT.sol#107-126) sends eth to
arbitrary user
        Dangerous calls:
        - address(msg.sender).transfer(address(this).balance) (contracts/
MtopNFT.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations

MtopNFT._getJSON(MtopNFT.Tier) (contracts/MtopNFT.sol#196-206) uses a dangerous strict
equality:
        - _tier == Tier._30 (contracts/MtopNFT.sol#197)
MtopNFT._getJSON(MtopNFT.Tier) (contracts/MtopNFT.sol#196-206) uses a dangerous strict
equality:
        - _tier == Tier._90 (contracts/MtopNFT.sol#199)
MtopNFT._getJSON(MtopNFT.Tier) (contracts/MtopNFT.sol#196-206) uses a dangerous strict
equality:
        - _tier == Tier._diamond (contracts/MtopNFT.sol#201)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

MtopNFT.constructor(IERC20,IOracle,IOracle,address)._feeCollector (contracts/
MtopNFT.sol#61) lacks a zero-check on :
                - feeCollector = _feeCollector (contracts/MtopNFT.sol#69)
MtopNFT.setFeeCollector(address)._newFeeCollector (contracts/MtopNFT.sol#76) lacks a
zero-check on :
                - feeCollector = _newFeeCollector (contracts/MtopNFT.sol#77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

Reentrancy in MtopNFT.buy(address,MtopNFT.Tier) (contracts/MtopNFT.sol#107-126):
        External calls:
        - MTOP.safeTransferFrom(msg.sender,feeCollector,_mtopAmount) (contracts/
MtopNFT.sol#115)
        External calls sending eth:
        - address(feeCollector).transfer(_oneAmount) (contracts/MtopNFT.sol#116)
        - address(msg.sender).transfer(address(this).balance) (contracts/
MtopNFT.sol#120)
```

```
        State variables written after the call(s):
        - nftDetails[tokenId] = NFTDetails(tier,_getExpirationTimestamp(tier))
(contracts/MtopNFT.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

MtopNFT.tokenURI(uint256) (contracts/MtopNFT.sol#171-186) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(details.expirationTimestamp != 0,MtopNFT: !tokenId)
(contracts/MtopNFT.sol#179)
MtopNFT._getJSON(MtopNFT.Tier) (contracts/MtopNFT.sol#196-206) uses timestamp for
comparisons
        Dangerous comparisons:
        - _tier == Tier._30 (contracts/MtopNFT.sol#197)
        - _tier == Tier._90 (contracts/MtopNFT.sol#199)
        - _tier == Tier._diamond (contracts/MtopNFT.sol#201)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

Decimals.mulWithPrecision(uint256,uint256,uint8) (contracts/libraries/
Decimals.sol#13-19) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.9 (contracts/interfaces/IOracle.sol#2) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/libraries/Decimals.sol#2) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Parameter MtopNFT.setFeeCollector(address)._newFeeCollector (contracts/MtopNFT.sol#76)
is not in mixedCase
Parameter MtopNFT.setOneOracle(IOracle)._newOneOracle (contracts/MtopNFT.sol#80) is not
in mixedCase
Parameter MtopNFT.setMtopOracle(IOracle)._newMtopOracle (contracts/MtopNFT.sol#84) is
not in mixedCase
Parameter MtopNFT.getTierCost(MtopNFT.Tier)._tier (contracts/MtopNFT.sol#128) is not in
mixedCase
Variable MtopNFT.SECONDS_PER_DAY (contracts/MtopNFT.sol#43) is not in mixedCase
```

```
Variable MtopNFT.MTOP (contracts/MtopNFT.sol#45) is not in mixedCase
Variable MtopNFT.MTOP_DECIMALS (contracts/MtopNFT.sol#46) is not in mixedCase
Variable MtopNFT.MTOP_ORACLE (contracts/MtopNFT.sol#48) is not in mixedCase
Variable MtopNFT.ONE_ORACLE (contracts/MtopNFT.sol#49) is not in mixedCase
Parameter Decimals.divWithPrecision(uint256,uint256,uint8)._numeratorAmount (contracts/
libraries/Decimals.sol#6) is not in mixedCase
Parameter Decimals.divWithPrecision(uint256,uint256,uint8)._denominatorAmount
(contracts/libraries/Decimals.sol#7) is not in mixedCase
Parameter Decimals.divWithPrecision(uint256,uint256,uint8)._precision (contracts/
libraries/Decimals.sol#8) is not in mixedCase
Parameter Decimals.mulWithPrecision(uint256,uint256,uint8)._amountA (contracts/
libraries/Decimals.sol#14) is not in mixedCase
Parameter Decimals.mulWithPrecision(uint256,uint256,uint8)._amountB (contracts/
libraries/Decimals.sol#15) is not in mixedCase
Parameter Decimals.mulWithPrecision(uint256,uint256,uint8)._precision (contracts/
libraries/Decimals.sol#16) is not in mixedCase
Parameter Decimals.formatFromToDecimals(uint8,uint8,uint256)._fromDecimals (contracts/
libraries/Decimals.sol#22) is not in mixedCase
Parameter Decimals.formatFromToDecimals(uint8,uint8,uint256)._toDecimals (contracts/
libraries/Decimals.sol#23) is not in mixedCase
Parameter Decimals.formatFromToDecimals(uint8,uint8,uint256)._amount (contracts/
libraries/Decimals.sol#24) is not in mixedCase
Parameter Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._decimalsA
(contracts/libraries/Decimals.sol#39) is not in mixedCase
Parameter Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._decimalsB
(contracts/libraries/Decimals.sol#40) is not in mixedCase
Parameter Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._amountA
(contracts/libraries/Decimals.sol#41) is not in mixedCase
Parameter Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._amountB
(contracts/libraries/Decimals.sol#42) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Variable MtopNFT.MTOP_ORACLE (contracts/MtopNFT.sol#48) is too similar to
MtopNFT.constructor(IERC20,IOracle,IOracle,address)._mtopOracle (contracts/
MtopNFT.sol#60)
Variable Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._amountAFormatted
(contracts/libraries/Decimals.sol#47) is too similar to
Decimals.formatToBiggerDecimals(uint8,uint8,uint256,uint256)._amountBFormatted
(contracts/libraries/Decimals.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
```

```
are-too-similar

MtopNFT.SECONDS_PER_DAY (contracts/MtopNFT.sol#43) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

batchGift(address[],MtopNFT.Tier) should be declared external:
        - MtopNFT.batchGift(address[],MtopNFT.Tier) (contracts/MtopNFT.sol#88-92)
buy(address,MtopNFT.Tier) should be declared external:
        - MtopNFT.buy(address,MtopNFT.Tier) (contracts/MtopNFT.sol#107-126)
baseURI() should be declared external:
        - MtopNFT.baseURI() (contracts/MtopNFT.sol#167-169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

0x Guard