QuillAudits

Audit Report
September, 2021

For

ELEMON

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the ELEMON smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	4	2	0
Closed	4	1	4	0

Introduction

During the period of September 06, 2021, to September 15, 2021 - QuillAudits Team performed a security audit for ELMON smart contracts.

The code for the audit was taken from the following official repo of ELEMON: https://github.com/ELMONgame/contracts

Note	Date	Commit hash
Version 1	September	1bcd31f92b8ce7cbcc113a39e80de3481d5e7ad0
Version 2	September	1f559e27074c199f6b0175a724eb4a145a1638d0

Issues Found

A. Contract – ERC20

High severity issues

No issues were found.

Medium severity issues

1. Approve Race

```
Line 83:
function approve(address spender, uint256 amount)
        public
        virtual
        override
        returns (bool)
        {
        _approve(_msgSender(), spender, amount);
        return true;
        }
```

Description

The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Remediation

Avoid using the approve function to overwrite the allowance value, use instead of the
increaseAllowance and decreaseAllowance functions.

Status: Acknowledged by the Auditee

2. Renounce Ownership

```
Line 39:
contract ERC20 is Ownable, IERC20, IERC20Metadata {
```

Description
Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Remediation
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status: Acknowledged by the Auditee

Low level severity issues

3. Floating pragma

pragma solidity ^0.8.0;

Description
The contract makes use of the floating-point pragma 0.8.0 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation
Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status: Fixed
The Element team has fixed the issue in version 2 by locking the pragma version.

4. Public Function That Can Be Declared External

Description

The following public functions that are never called by the contract should be declared external to save gas:

- name()
- symbol()
- decimals()
- totalSupply()
- balanceOf()
- transfer()
- allowance()
- approve()
- transferFrom()
- increaseAllowance()
- decreaseAllowance()

Remediation

Use the external attribute for functions that are not called from the contract.

Status: Acknowledged by the Auditee

5. Unnecessary unchecked call

```
Line 104:
    require(currentAllowance >= amount,
      "ERC20: transfer amount exceeds allowance");
       unchecked {
            _approve(sender, _msgSender(), currentAllowance - amount);
       }
```

```
Line 135:
    require(currentAllowance >= subtractedValue,
            "ERC20: decreased allowance below zero");
unchecked {
            _approve(sender, _msgSender(), currentAllowance - amount);
       }
```

```
Line 157:
    require(senderBalance >= amount,
            "ERC20: transfer amount exceeds balance");
        unchecked {
            _balances[sender] = senderBalance - amount;
}
```

```
Line 190:
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
}
```

Description

In some functions the unchecked option is used to allow overflows/underflows and to disable the built-in overflow/underflow protection that solidity 0.8.0 and latest versions implement. But the values that are inside the unchecked block are obtained as a result of subtraction that will never underflow due to the fact that the previous require instruction forces the first number to be greater than the second one.

Remediation

Remove the unchecked block since it is not achieving any objective in the smart contract's logic.

Status: Acknowledged by the Auditee

B. Contract - ELMONMarketplace

High severity issues

1. Missing Value Verification

```
Line 221:
function setFeePercent(uint256 feePercent) external onlyOwner {
            _feePercent = feePercent;
        }
```

Description

Certain functions lack a safety check in the value, the feePercent argument should be lower than 100000. Otherwise, the contract's logic might be harmed.

Remediation

Add a require or modifier in order to verify that the input value is lower than 100000.

Status: Fixed

The ELEMON team has fixed the issue by adding a require that verifies that the input value is lower than 100 * MULTIPLIER.

2. Race condition on FeePercent Variable

```
Line 221:
function setFeePercent(uint256 feePercent) external onlyOwner {
            _feePercent = feePercent;
        }
```

Description

In the contract the user can call the function getFeePercent to check the _feePercent then he might decide to purchase an ELMONNFT but at the same time the owner might have called setFeePercent to modify the fee. In the scenario where the owner's transaction gets mined first the user could possibly buy tokens with a different price than the one that was returned by getTokenPrice.

Remediation

Add the feePercent as an additional parameter to the purchase function and add a require condition that verifies that the fee provided in the parameters is equal to the current feePercent in the smart contract.

Status: Fixed

The Element team has fixed the issue in version 2 by adding the feePercent as a new parameter in the purchase function, in addition to that, they added a require that verifies that this parameter is the same as the one stored in the smart contract.

3. Race condition – Forcing User to Buy with Higher Price

```
Line 152:
function purchase(uint256 tokenId) external returns (uint256) {
            address tokenOwner = _tokenOwners[tokenId];
          require(tokenOwner != address(0), "Token has not been added");
          uint256 tokenPrice = _tokenPrices[tokenId];
         if (tokenPrice > 0) {
              IERC20 ELMONTokenContract = IERC20(_ELMONTokenAddress);
               require(ELMONTokenContract.transferFrom(
                _msgSender(),
               address(this),
                     tokenPrice));
               uint256 feeAmount = 0;
               if (_feePercent > 0) {
                  feeAmount = (tokenPrice * _feePercent)/100/ MULTIPLIER;
                  require(ELMONTokenContract.transfer(
                                owner(),feeAmount));
               }
                  require(ELMONTokenContract.transfer(
                           tokenOwner, tokenPrice - feeAmount)
                  );
          }
```

Description

The user can create a sell order then cancel it and create it once again with a higher price. If someone purchased the token before the cancel and his transaction got mined after the create and cancel transactions, he will purchase the token with a higher price than the one he saw before the operation.

Remediation

Add the tokenPrice as an additional parameter to the purchase function and add a require condition that verifies that the price provided in the parameters is equal to the current tokenPrice in the smart contract.

Status: Fixed

The Element team has fixed the issue in version 2 by adding the requestTokenPrice as a new parameter in the purchase function, in addition to that, they added a require that verifies that this parameter is the same as the one stored in the smart contract.

Medium severity issues

4. Loop over a dynamic array

```
Line 228:
function _removeFromTokens(uint256 tokenId) internal view returns (uint256[] memory) {
        uint256 tokenCount = _tokens.length;
        uint256[] memory result = new uint256[](tokenCount - 1);
        uint256 resultIndex = 0;
        for (uint256 tokenIndex = 0; tokenIndex < tokenCount; tokenIndex++) {
            uint256 tokenItemId = _tokens[tokenIndex];
            if (tokenItemId != tokenId) {
                result[resultIndex] = tokenItemId;
                resultIndex++;
            }
        }
        return result;
    }
```

Description

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Remediation
Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status: Fixed
The Element team has fixed the issue in version 2 by removing the arrays and loops from the logic from the smart contract.

Low level severity issues

5. Floating pragma

pragma solidity ^0.8.0;

Description
The contract makes use of the floating-point pragma 0.8.0 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensuring that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation
Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status: Fixed
The Element team has fixed the issue in version 2 by locking the pragma version.

6. Usage of block.timestamp

```
Line 221:MarketHistory({
_marketHistories[tokenId].push(

            buyer: _msgSender(),
            seller: _tokenOwners[tokenId],
            price: tokenPrice,
            time: block.timestamp
        })
    );
```

Description
Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right, all what is guaranteed is that it is higher than the timestamp of the previous block.

Remediation
You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't destroy the logic of the staking contract.

Status: Fixed
The Element team has fixed in version 2 the issue by removing the timestamps from the logic of the smart contracts.

C. Contract - ELMONNFT

High severity issues

No issues were found.

Medium severity issues

1. **Loop over dynamic array**

```
Line 294:
function toHexString(uint256 value) internal pure returns (string memory)      {
            if (value == 0) {
                return "0x00";
            }
            uint256 temp = value;
            uint256 length = 0;
            while (temp != 0) {
                length++;
                temp >>= 8;
            }
            return toHexString(value, length);
        }
```

```
Line 314:
function toHexString(uint256 value, uint256 length) internal pure  returns (string memory)
        {
            bytes memory buffer = new bytes(2 * length + 2);
            buffer[0] = "0";
            buffer[1] = "x";
            for (uint256 i = 2 * length + 1; i > 1; --i) {
                buffer[i] = alphabet[value & 0xf];
                value >>= 4;
            }
            require(value == 0, "Strings: hex length insufficient");
            return string(buffer);
        }
```

```
Line 838:
function mintMultiples(address to, uint256[] memory tokenIds) public
        onlyOwner
        {
            require(tokenIds.length > 0, "Empty array");
            for (uint256 index = 0; index < tokenIds.length; index++)
          _safeMint(to, tokenIds[index]);
        }
```

Description

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial of Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Remediation

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status: Acknowledged by the Auditee

Low level severity issues

No issues were found.

D. Contract - ELMONShop

High severity issues

1. Race condition – The Token Price

```
Line 108:
function purchase(uint256 tokenId) public payable whenRunning returns (bool)
        {
            require(!_isSold[tokenId], "Sold");
            uint256 price = getTokenPrice(tokenId);
            require(price > 0, "Price is 0");
            require(msg.value == price, "Invalid BNB to purchase");
            _ELMONNft.safeTransferFrom(address(this), _msgSender(), tokenId);
            _isSold[tokenId] = true;
            emit Purchased(_msgSender(), tokenId, price, _now());
            return true;
        }
```

Description

In the contract the user can call the function getTokenPrice to check the price then he might decide to buy the NFT token but at the same time the owner might have called setStarPrice to modify the price. In the scenario where the owner's transaction gets mined first the user could possibly buy the token with a different price than the one that was returned by getTokenPrice.

Remediation

Add the token price as an additional parameter to the purchase function and add a require condition that verifies that the price provided in the parameters is equal to the current price in the smart contract.

Status: Fixed

The ELEMON team has fixed the issue in version 2 by adding the requestPrice parameter as an argument in the purchase function, in addition, to require that verifies that the parameter is the same as the one stored in the smart contract.

Medium severity issues

2. For loop over over dynamic array

```
Line 58:
function setStarPriceMultiple(uint256[] memory stars, uint256[] memory prices) public
onlyOwner {
        require(stars.length > 0, "Empty array");
        require(stars.length == prices.length, "Invalid input");
        for (uint256 index = 0; index < stars.length; index++) {
            require(stars[index] > 0, "Star is 0");
            require(prices[index] > 0, "Price is 0");
            _starPrices[stars[index]] = prices[index];
        }
    }
```

```
Line 86:
    function setTokenStarMultiple(
        uint256[] memory tokenIds,
        uint256[] memory stars
        ) public onlyOwner {
        require(tokenIds.length > 0, "Empty array");
        require(tokenIds.length == stars.length, "Invalid input");
        for (uint256 index = 0; index < stars.length; index++) {
            require(stars[index] > 0, "Star is 0");
            _tokenStars[tokenIds[index]] = stars[index];
        }
    }
```

Description

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Remediation

It is recommended to add a require or modifier in order to verify that the input address is different than the zero-address.

Status: Acknowledged by the Auditee

Low level severity issues

3. Floating pragma

pragma solidity ^0.8.0;

Description
The contract makes use of the floating-point pragma 0.8.0 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensuring that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation
Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status: Fixed
The Element team has fixed the issue in version 2 by locking the pragma version.

Functional test

Function Names	Testing results
withdrawToken	Passed
mint	Passed
mintMultiples	Passed
burn	Passed
supportsInterface	Passed
balanceOf	Passed
ownerOf	Passed
name	Passed
symbol	Passed
tokenURI	Passed
approve	Passed
getApproved	Passed
setApprovalForAll	Passed
isApprovedForAll	Passed
transferFrom	Passed
safeTransferFrom	Passed
setELMONNft	Passed
setStarPrice	Passed
setTokenStar	Passed

Function Names	Testing results
setTokenStarMultiple	Passed
getTokenPrice	Passed
purchase	Passed
withdrawToken	Passed
setNftContractOwner	Passed
withdrawBnb	Passed

Automated Tests

Slither

```
INFO:Detectors:
Pragma version^0.8.0 (ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
supportsInterface(bytes4) should be declared external:
        - ERC165.supportsInterface(bytes4) (ERC165.sol#11-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
ElemonShop.withdrawBnb() (ElemonShop.sol#121-124) sends eth to arbitrary user
        Dangerous calls:
        - address(owner()).transfer(address(this).balance) (ElemonShop.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
ElemonShop.withdrawToken(address) (ElemonShop.sol#109-112) ignores return value by token.transfer(owner(),token.balanceOf(address(this))) (ElemonShop.sol#111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in ElemonShop.purchase(uint256) (ElemonShop.sol#93-104):
        External calls:
        - _elemonNft.safeTransferFrom(address(this),_msgSender(),tokenId) (ElemonShop.sol#99)
        State variables written after the call(s):
        - _isSold[tokenId] = true (ElemonShop.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Reentrancy in ElemonShop.purchase(uint256) (ElemonShop.sol#93-104):
        External calls:
        - _elemonNft.safeTransferFrom(address(this),_msgSender(),tokenId) (ElemonShop.sol#99)
        Event emitted after the call(s):
        - Purchased(_msgSender(),tokenId,price,_now()) (ElemonShop.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (utils/Context.sol#13-16) 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 (ElemonShop.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Ownable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Runnable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable ElemonShop._elemonNft (ElemonShop.sol#17) is not in mixedCase
Variable ElemonShop._isSold (ElemonShop.sol#20) is not in mixedCase
Variable ElemonShop._starPrices (ElemonShop.sol#23) is not in mixedCase
Variable ElemonShop._tokenStars (ElemonShop.sol#26) is not in mixedCase
Variable Runnable._isRunning (utils/Runnable.sol#18) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (utils/Context.sol#14)" inContext (utils/Context.sol#4-22)
Redundant expression "this (utils/Context.sol#19)" inContext (utils/Context.sol#4-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
setElemonNft(address) should be declared external:
        - ElemonShop.setElemonNft(address) (ElemonShop.sol#32-35)
setStarPrice(uint256,uint256) should be declared external:
        - ElemonShop.setStarPrice(uint256,uint256) (ElemonShop.sol#41-45)
setStarPriceMultiple(uint256[],uint256[]) should be declared external:
        - ElemonShop.setStarPriceMultiple(uint256[],uint256[]) (ElemonShop.sol#51-60)
```

```
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#60-74)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20.sol#76-79)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20.sol#81-89)
owner() should be declared external:
        - Ownable.owner() (utils/Ownable.sol#21-23)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (utils/Ownable.sol#40-43)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (utils/Ownable.sol#49-51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
ElemonMarketplace.withdrawToken(address) (ElemonMarketplace.sol#211-214) ignores return value by token.transfer(owner(),token.balanceOf(address(this))) (
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in ElemonMarketplace.cancelSellOrder(uint256) (ElemonMarketplace.sol#67-81):
        External calls:
        - elemonContract.safeTransferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#72)
        State variables written after the call(s):
        - _tokenOwners[tokenId] = address(0) (ElemonMarketplace.sol#74)
Reentrancy in ElemonMarketplace.createSellOrder(uint256,uint256) (ElemonMarketplace.sol#45-61):
        External calls:
        - elemonContract.safeTransferFrom(_msgSender(),address(this),tokenId) (ElemonMarketplace.sol#52)
        State variables written after the call(s):
        - _tokenOwners[tokenId] = _msgSender() (ElemonMarketplace.sol#54)
Reentrancy in ElemonMarketplace.purchase(uint256) (ElemonMarketplace.sol#131-165):
        External calls:
        - require(bool)(elemonTokenContract.transferFrom(_msgSender(),address(this),tokenPrice)) (ElemonMarketplace.sol#139)
        - require(bool)(elemonTokenContract.transfer(owner(),feeAmount)) (ElemonMarketplace.sol#143)
        - require(bool)(elemonTokenContract.transfer(tokenOwner,tokenPrice - feeAmount)) (ElemonMarketplace.sol#145)
        - IERC721(_elemonNftAddress).transferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#149)
        State variables written after the call(s):
        - _tokenOwners[tokenId] = address(0) (ElemonMarketplace.sol#158)
        - _tokenPrices[tokenId] = 0 (ElemonMarketplace.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ElemonMarketplace.constructor(address,address).tokenAddress (ElemonMarketplace.sol#34) lacks a zero-check on :
                - _elemonTokenAddress = tokenAddress (ElemonMarketplace.sol#35)
ElemonMarketplace.constructor(address,address).nftAddress (ElemonMarketplace.sol#34) lacks a zero-check on :
                - _elemonNftAddress = nftAddress (ElemonMarketplace.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ElemonMarketplace.cancelSellOrder(uint256) (ElemonMarketplace.sol#67-81):
        External calls:
        - elemonContract.safeTransferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#72)
        State variables written after the call(s):
        - _tokenPrices[tokenId] = 0 (ElemonMarketplace.sol#75)
        - _tokens = _removeFromTokens(tokenId) (ElemonMarketplace.sol#76)
Reentrancy in ElemonMarketplace.createSellOrder(uint256,uint256) (ElemonMarketplace.sol#45-61):
        External calls:
        - elemonContract.safeTransferFrom(_msgSender(),address(this),tokenId) (ElemonMarketplace.sol#52)
        State variables written after the call(s):
        - _tokenPrices[tokenId] = price (ElemonMarketplace.sol#55)
        - _tokens.push(tokenId) (ElemonMarketplace.sol#56)
Reentrancy in ElemonMarketplace.purchase(uint256) (ElemonMarketplace.sol#131-165):
        External calls:
        - require(bool)(elemonTokenContract.transferFrom(_msgSender(),address(this),tokenPrice)) (ElemonMarketplace.sol#139)
        - require(bool)(elemonTokenContract.transfer(owner(),feeAmount)) (ElemonMarketplace.sol#143)
        - require(bool)(elemonTokenContract.transfer(tokenOwner,tokenPrice - feeAmount)) (ElemonMarketplace.sol#145)
        - IERC721(_elemonNftAddress).transferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#149)
        State variables written after the call(s):
        - _marketHistories[tokenId].push(MarketHistory(_msgSender(),_tokenOwners[tokenId],tokenPrice,block.timestamp)) (ElemonMarketplace.sol#151-156)
```

```
setTokenStar(uint256,uint256) should be declared external:
        - ElemonShop.setTokenStar(uint256,uint256) (ElemonShop.sol#66-69)
setTokenStarMultiple(uint256[],uint256[]) should be declared external:
        - ElemonShop.setTokenStarMultiple(uint256[],uint256[]) (ElemonShop.sol#75-83)
purchase(uint256) should be declared external:
        - ElemonShop.purchase(uint256) (ElemonShop.sol#93-104)
withdrawToken(address) should be declared external:
        - ElemonShop.withdrawToken(address) (ElemonShop.sol#109-112)
setNftContractOwner(address) should be declared external:
        - ElemonShop.setNftContractOwner(address) (ElemonShop.sol#114-116)
withdrawBnb() should be declared external:
        - ElemonShop.withdrawBnb() (ElemonShop.sol#121-124)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (utils/Ownable.sol#40-43)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (utils/Ownable.sol#49-51)
toggleRunning() should be declared external:
        - Runnable.toggleRunning() (utils/Runnable.sol#24-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
ERC20.allowance(address,address).owner (ERC20.sol#51) shadows:
        - Ownable.owner() (utils/Ownable.sol#21-23) (function)
ERC20._approve(address,address,uint256).owner (ERC20.sol#143) shadows:
        - Ownable.owner() (utils/Ownable.sol#21-23) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (utils/Context.sol#13-16) is never used and should be removed
Context._now() (utils/Context.sol#18-21) is never used and should be removed
ERC20._burn(address,uint256) (ERC20.sol#125-140) 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 (ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Ownable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Redundant expression "this (utils/Context.sol#14)" inContext (utils/Context.sol#4-22)
Redundant expression "this (utils/Context.sol#19)" inContext (utils/Context.sol#4-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable ERC20._totalSupply (ERC20.sol#14) is too similar to ERC20.constructor(string,string,uint256).totalSupply_ (ERC20.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (ERC20.sol#26-28)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#30-32)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#34-36)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20.sol#38-40)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#42-44)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#46-49)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20.sol#51-53)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#55-58)
transferFrom(address,address,uint256) should be declared external:
```

```
        - _tokens = _removeFromTokens(tokenId) (ElemonMarketplace.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ElemonMarketplace.cancelSellOrder(uint256) (ElemonMarketplace.sol#67-81):
        External calls:
        - elemonContract.safeTransferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#72)
        Event emitted after the call(s):
        - SellingOrderCanceled(tokenId,_now()) (ElemonMarketplace.sol#78)
Reentrancy in ElemonMarketplace.createSellOrder(uint256,uint256) (ElemonMarketplace.sol#45-61):
        External calls:
        - elemonContract.safeTransferFrom(_msgSender(),address(this),tokenId) (ElemonMarketplace.sol#52)
        Event emitted after the call(s):
        - NewSellOrderCreated(_msgSender(),tokenId,price,_now()) (ElemonMarketplace.sol#58)
Reentrancy in ElemonMarketplace.purchase(uint256) (ElemonMarketplace.sol#131-165):
        External calls:
        - require(bool)(elemonTokenContract.transferFrom(_msgSender(),address(this),tokenPrice)) (ElemonMarketplace.sol#139)
        - require(bool)(elemonTokenContract.transfer(owner(),feeAmount)) (ElemonMarketplace.sol#143)
        - require(bool)(elemonTokenContract.transfer(tokenOwner,tokenPrice - feeAmount)) (ElemonMarketplace.sol#145)
        - IERC721(_elemonNftAddress).transferFrom(address(this),_msgSender(),tokenId) (ElemonMarketplace.sol#149)
        Event emitted after the call(s):
        - Purchased(_msgSender(),tokenId,tokenPrice,_now()) (ElemonMarketplace.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (utils/Context.sol#13-16) 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 (ElemonMarketplace.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Ownable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable ElemonMarketplace._elemonTokenAddress (ElemonMarketplace.sol#18) is not in mixedCase
Variable ElemonMarketplace._elemonNftAddress (ElemonMarketplace.sol#19) is not in mixedCase
Variable ElemonMarketplace._tokens (ElemonMarketplace.sol#24) is not in mixedCase
Variable ElemonMarketplace._tokenPrices (ElemonMarketplace.sol#27) is not in mixedCase
Variable ElemonMarketplace._tokenOwners (ElemonMarketplace.sol#30) is not in mixedCase
Variable ElemonMarketplace._marketHistories (ElemonMarketplace.sol#32) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (utils/Context.sol#14)" inContext (utils/Context.sol#4-22)
Redundant expression "this (utils/Context.sol#19)" inContext (utils/Context.sol#4-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
withdrawToken(address) should be declared external:
        - ElemonMarketplace.withdrawToken(address) (ElemonMarketplace.sol#211-214)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (utils/Ownable.sol#40-43)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (utils/Ownable.sol#49-51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
Pragma version^0.8.0 (Migrations.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
ERC20.allowance(address,address).owner (ERC20.sol#51) shadows:
        - Ownable.owner() (utils/Ownable.sol#21-23) (function)
ERC20._approve(address,address,uint256).owner (ERC20.sol#143) shadows:
        - Ownable.owner() (utils/Ownable.sol#21-23) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (utils/Context.sol#13-16) is never used and should be removed
Context._now() (utils/Context.sol#18-21) is never used and should be removed
ERC20._burn(address,uint256) (ERC20.sol#125-140) 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 (ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (ElemonToken.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Ownable.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Redundant expression "this (utils/Context.sol#14)" inContext (utils/Context.sol#4-22)
Redundant expression "this (utils/Context.sol#19)" inContext (utils/Context.sol#4-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable ERC20._totalSupply (ERC20.sol#14) is too similar to ERC20.constructor(string,string,uint256).totalSupply_ (ERC20.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ElemonToken.constructor() (ElemonToken.sol#6) uses literals with too many digits:
        - ERC20(Elemon Token,EMON,500000000000000000000000000) (ElemonToken.sol#6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (ERC20.sol#26-28)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#30-32)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#34-36)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20.sol#38-40)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#42-44)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#46-49)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20.sol#51-53)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#55-58)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#60-74)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20.sol#76-79)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20.sol#81-89)
owner() should be declared external:
        - Ownable.owner() (utils/Ownable.sol#21-23)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (utils/Ownable.sol#40-43)
transferOwnership(address) should be declared external:
```

```
        - Ownable.transferOwnership(address) (utils/Ownable.sol#49-51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
ERC721._checkOnERC721Received(address,address,uint256,bytes) (ElemonNFT.sol#599-618) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (ElemonNFT.sol#603-614)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (ElemonNFT.sol#603)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (ElemonNFT.sol#599-618) potentially used before declaration: retval == IE
C721Receiver(to).onERC721Received.selector (ElemonNFT.sol#604)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (ElemonNFT.sol#605)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (ElemonNFT.sol#599-618) potentially used before declaration: reason.lengt
 == 0 (ElemonNFT.sol#606)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (ElemonNFT.sol#605)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (ElemonNFT.sol#599-618) potentially used before declaration: revert(uint2
6,uint256)(32 + reason,mload(uint256)(reason)) (ElemonNFT.sol#611)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (ElemonNFT.sol#28-37) uses assembly
        - INLINE ASM (ElemonNFT.sol#35)
Address._verifyCallResult(bool,bytes,string) (ElemonNFT.sol#173-190) uses assembly
        - INLINE ASM (ElemonNFT.sol#182-185)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (ElemonNFT.sol#599-618) uses assembly
        - INLINE ASM (ElemonNFT.sol#610-612)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (ElemonNFT.sol#173-190) is never used and should be removed
Address.functionCall(address,bytes) (ElemonNFT.sol#81-83) is never used and should be removed
Address.functionCall(address,bytes,string) (ElemonNFT.sol#91-93) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ElemonNFT.sol#106-108) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ElemonNFT.sol#116-123) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ElemonNFT.sol#155-157) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ElemonNFT.sol#165-171) is never used and should be removed
Address.functionStaticCall(address,bytes) (ElemonNFT.sol#131-133) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (ElemonNFT.sol#141-147) is never used and should be removed
Address.sendValue(address,uint256) (ElemonNFT.sol#55-61) is never used and should be removed
Context._msgData() (utils/Context.sol#13-16) is never used and should be removed
Context._now() (utils/Context.sol#18-21) is never used and should be removed
Strings.toHexString(uint256) (ElemonNFT.sol#224-235) is never used and should be removed
Strings.toHexString(uint256,uint256) (ElemonNFT.sol#240-250) 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 (ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (ElemonNFT.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (interfaces/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
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) (ElemonNFT.sol#55-61):
        - (success) = recipient.call{value: amount}() (ElemonNFT.sol#59)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ElemonNFT.sol#116-123):
        - (success,returndata) = target.call{value: value}(data) (ElemonNFT.sol#121)
Low level call in Address.functionStaticCall(address,bytes,string) (ElemonNFT.sol#141-147):
        - (success,returndata) = target.staticcall(data) (ElemonNFT.sol#145)
Low level call in Address.functionDelegateCall(address,bytes,string) (ElemonNFT.sol#165-171):
        - (success,returndata) = target.delegatecall(data) (ElemonNFT.sol#169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Constant Strings.alphabet (ElemonNFT.sol#194) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (ElemonNFT.sol#428) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (utils/Context.sol#14)" inContext (utils/Context.sol#4-22)
```

```
Redundant expression "this (utils/Context.sol#19)" inContext (utils/Context.sol#4-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (ElemonNFT.sol#305-308)
name() should be declared external:
        - ERC721.name() (ElemonNFT.sol#333-335)
symbol() should be declared external:
        - ERC721.symbol() (ElemonNFT.sol#340-342)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (ElemonNFT.sol#347-354)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (ElemonNFT.sol#367-376)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (ElemonNFT.sol#394-399)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (ElemonNFT.sol#411-416)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (ElemonNFT.sol#421-423)
mint(address,uint256) should be declared external:
        - ElemonNFT.mint(address,uint256) (ElemonNFT.sol#643-645)
mintMultiples(address,uint256[]) should be declared external:
        - ElemonNFT.mintMultiples(address,uint256[]) (ElemonNFT.sol#647-651)
burn(uint256) should be declared external:
        - ElemonNFT.burn(uint256) (ElemonNFT.sol#653-656)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (40 contracts with 75 detectors), 179 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Mythx

Report for ElemonShop.sol
https://dashboard.mythx.io/#/console/analyses/952cc230-e50e-4df8-8512-b43a67ec5995

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
16	(SWC-123) Requirement Violation	Low	Requirement violation.
111	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for utils/Context.sol
https://dashboard.mythx.io/#/console/analyses/b3878905-c0a1-4e9b-ae5c-2846f1cf8b2c

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for ERC20.sol
https://dashboard.mythx.io/#/console/analyses/28647f94-3f9d-4387-a7b2-a351fe3a69df

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for ElemonMarketplace.sol
https://dashboard.mythx.io/#/console/analyses/c29d545d-9697-4a84-a181-9925a5632265

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
10	(SWC-123) Requirement Violation	Low	Requirement violation.
213	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for ElemonToken.sol
https://dashboard.mythx.io/#/console/analyses/3dbcfe5b-777b-4196-9685-0024c64e6b1b

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for ElemonNFT.sol
https://dashboard.mythx.io/#/console/analyses/5c67c668-b8ca-4b37-af96-8d7c08681272

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
603	(SWC-123) Requirement Violation	Low	Requirement violation.
640	(SWC-123) Requirement Violation	Low	Requirement violation.

```
Report for Ownable.sol
https://dashboard.mythx.io/#/console/analyses/42a902f4-4755-4f6f-b2c4-157dcae63228
```

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

```
Report for Runnable.sol
https://dashboard.mythx.io/#/console/analyses/7eed6f66-64d9-48f2-a66c-d1836a327bad
```

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.

Many issues were discovered during the initial audit; all these vulnerabilities are fixed by the ELEMON Team.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the ELEMON Contracts. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the ELEMON Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report September, 2021

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com