QuillAudits

# Audit Report September, 2022

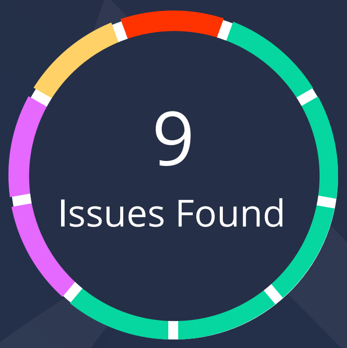
For

EDAO

# Table of Content

# Executive Summary

| | |
|---|---|
| **Project Name** | eDao |
| **Timeline** | September 5th, 2022 to September 8th, 2022 |
| **Method** | Manual Review, Functional Testing, Automated Testing. |
| **Scope of Audit** | The scope of this audit was to analyze eDao smart contract codebase for quality, security, and correctness. |
| **Commit Hash** | Source Code: (Contract 2 is under Audit Scope,Taken from Below Notion link provided by eDao Team) |
| **Fixed In** | *https://www.notion.so/dehidden/eDAO-Strands-Updated-Contract_QuillAudits-9b92a67576c14d5294611f375644106b* |



🟥 High 🟨 Medium

🟩 Low 🟪 Informational

| | High | Medium | Low | Informational |
|---|---|---|---|---|
| Open Issues | 0 | 0 | 0 | 0 |
| Acknowledged Issues | 1 | 1 | 3 | 0 |
| Partially Resolved Issues | 0 | 0 | 1 | 0 |
| Resolved Issues | 0 | 0 | 1 | 2 |

## Types of Severities

### 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 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.

## Types of Issues

### Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

### Resolved
These are the issues identified in the initial audit and have been successfully fixed.

### Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

### Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

# Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

## 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

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

# Manual Testing

## A. Contract - Strands.sol (2)

### High Severity Issues

#### A1. BaseURI can be changed anytime by admin

**Description**

Mutable URIs are not recommended for NFTs as these can be exploited by a malicious actor or compromised admin. All NFTs can be wiped out if baseURI is changed by a malicious admin.

**Remediation**

Recommended to follow immutable uris for NFTs

**eDao Team Comment**: Confirmation from client that the Base URI needs to be upgradeable to have more flexibility in link security.

**Status**

**Acknowledged**

### Medium Severity Issues

#### A2. Centralization risk

**Description**

All access control belongs to 'onlyOwner' and could lead to a DoS if owner account is compromised. If renounceOwnership function is called, the contract would lose major functionality. TokenURI, baseMetadataAPI, and addAdmin functions would be void.

**Remediation**

Ownable is not recommended for production, rather implement AccessControl contract instead. If Ownable must be used, a multi-sig is recommended.

**Status**

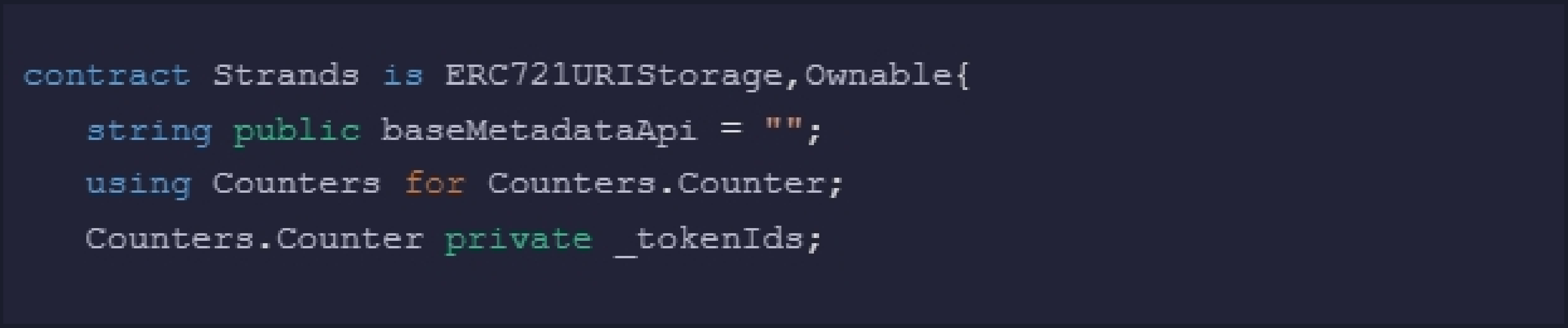
**Acknowledged**

## Low Severity Issues

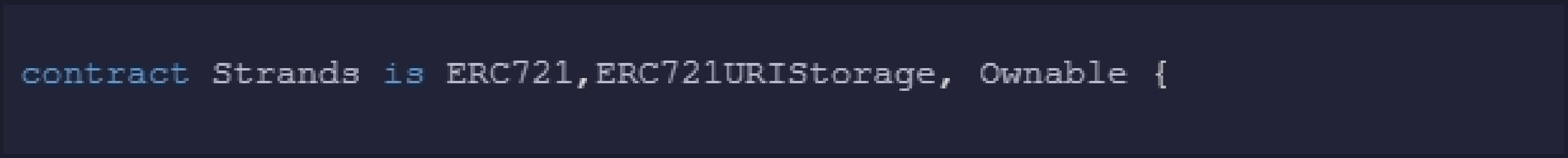
### A3. Contract naming convention

**Description**

Both contracts are named “Strands.sol”



```
contract Strands is ERC721URIStorage,Ownable{
    string public baseMetadataApi = "";
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;
```



```
contract Strands is ERC721,ERC721URIStorage, Ownable {
```

**Remediation**

Recommended to follow immutable uris for NFTs

**eDao Team Comment**: Confirmation from client that the Base URI needs to be upgradeable to have more flexibility in link security.

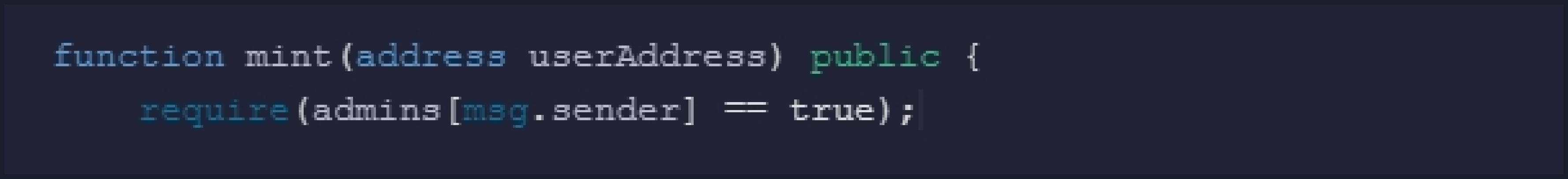
**Status**

**Acknowledged**

## A4. Comparison not required

**Description**

The comparison in the mint function is not necessary and costs extra gas.



```
function mint(address userAddress) public {
    require(admins[msg.sender] == true);
```

**Recommendation**

Change “(admins[msg.sender] == true)” to “(admins[msg.sender])” to optimize contract and save gas, also include an error message in the require statement.

**Status**

**Partially Resolved**

## A.5 Misleading function name

**Description**

The function “setPublicMintActive” makes it seem as though public mint functionality is included in the contract, but it isn’t, rather it sets the value for the bool “transferable” to make NFTs soulbound or no

**Recommendation**

Consider renaming function.

**Status**

**Resolved**

## A.6 Mints without baseMetadataAPI set

**Description**

When the mint function is called, the baseMetadataAPI is not checked if set or not. Although it can be updated later on with setTokenURI, ideally this shouldn’t happen.

**Recommendation**

Have require checks in place for baseMetadataAPI to be set before minting occurs.

**Status**

**Acknowledged**

## A.7 Minting begins at tokenId 0, not 1

**Description**

When the mint function is called, the first token to be minted has a tokenId of 0.

**Recommendation**

Allow minting to begin at tokenId 1 instead, else it could be inconsistent with the expected total supply of token.

**Status**

**Acknowledged**

## Informational Issues

### A.8 Naming convention

**Description**

Following the solidity naming convention for variables, enabletransfer should be written in camelcase.

**Remediation**

Change enabletransfer to enableTransfer

**Status**

**Resolved**

### A.9 Floating pragma used

**Description**

Pragma solidity version used here is ^0.8.4

**Remediation**

Recommended to lock the pragma version and consider known bugs for the chosen pragma version when deploying contracts.

**Status**

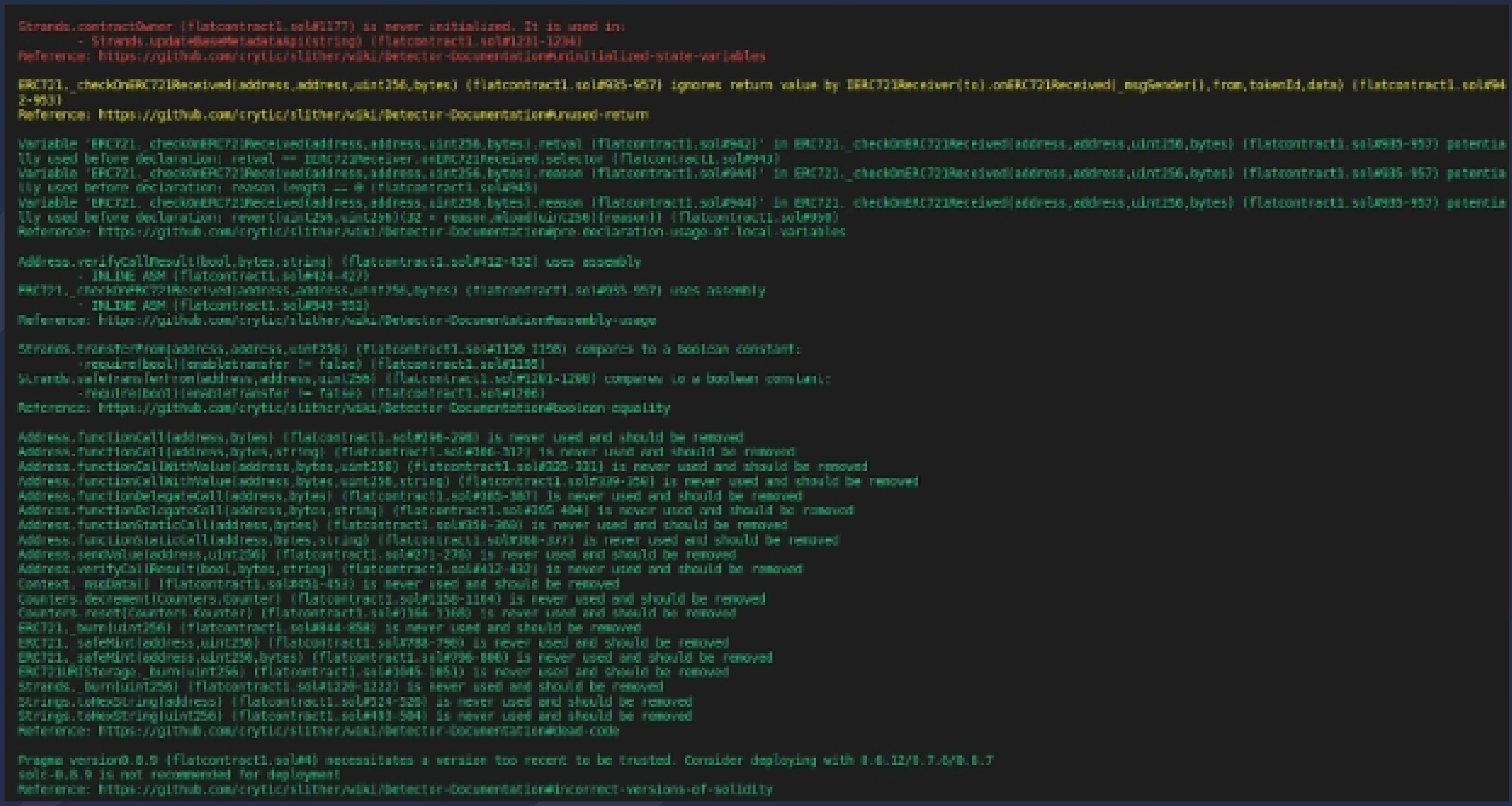
**Resolved**

## Functional Testing

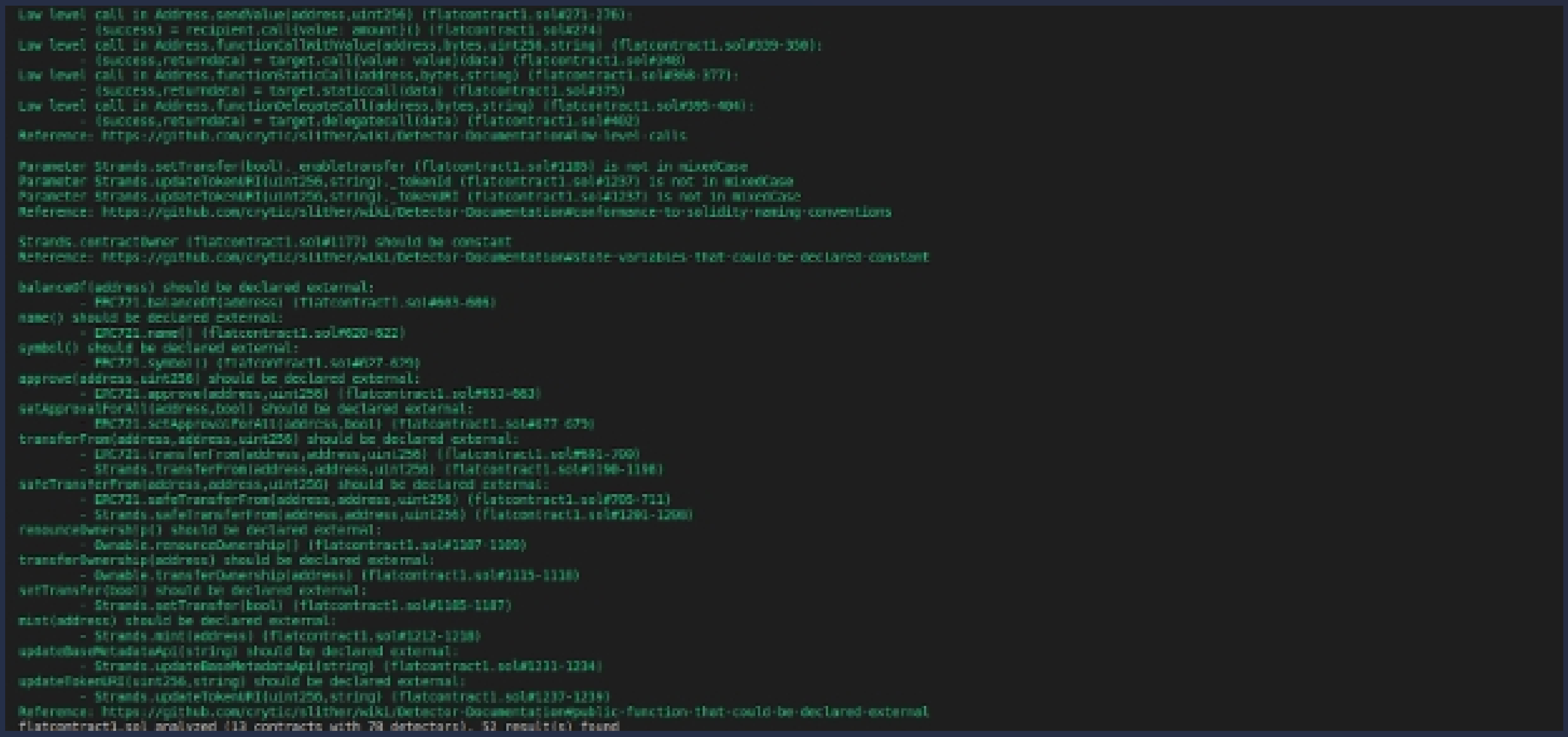
- ✓ Should fail on mint by non-admin
- ✓ Should mint when non-admin calls
- ✓ Transfers should revert when NFT is soulbound
- ✓ Should not mint when not admin
- ✗ Should mint starting from tokenId ‘1’
- ✗ Should mint with baseMetadataApi set initially

## Automated Tests

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



```
Strands.contractOwner (flatcontract1.sol#1177) is never initialized. It is used in:
        - Strands.updateBaseMetadataApi(string) (flatcontract1.sol#1231-1234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (flatcontract1.sol#935-957) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (flatcontract1.sol#94
2-953)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (flatcontract1.sol#942)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (flatcontract1.sol#935-957) potentia
lly used before declaration: retval == IERC721Receiver.onERC721Received.selector (flatcontract1.sol#943)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (flatcontract1.sol#944)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (flatcontract1.sol#935-957) potentia
lly used before declaration: reason.length == 0 (flatcontract1.sol#945)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (flatcontract1.sol#944)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (flatcontract1.sol#935-957) potentia
lly used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (flatcontract1.sol#950)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.verifyCallResult(bool,bytes,string) (flatcontract1.sol#412-432) uses assembly
        - INLINE ASM (flatcontract1.sol#424-427)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (flatcontract1.sol#935-957) uses assembly
        - INLINE ASM (flatcontract1.sol#949-951)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Strands.transferFrom(address,address,uint256) (flatcontract1.sol#1150-1158) compares to a boolean constant:
        -require(bool)(enabletransfer != false) (flatcontract1.sol#1155)
Strands.safeTransferFrom(address,address,uint256) (flatcontract1.sol#1201-1208) compares to a boolean constant:
        -require(bool)(enabletransfer != false) (flatcontract1.sol#1206)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Address.functionCall(address,bytes) (flatcontract1.sol#296-298) is never used and should be removed
Address.functionCall(address,bytes,string) (flatcontract1.sol#306-317) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (flatcontract1.sol#325-331) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (flatcontract1.sol#339-350) is never used and should be removed
Address.functionDelegateCall(address,bytes) (flatcontract1.sol#385-387) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (flatcontract1.sol#395-404) is never used and should be removed
Address.functionStaticCall(address,bytes) (flatcontract1.sol#358-360) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (flatcontract1.sol#368-377) is never used and should be removed
Address.sendValue(address,uint256) (flatcontract1.sol#271-276) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (flatcontract1.sol#412-432) is never used and should be removed
Context._msgData() (flatcontract1.sol#451-453) is never used and should be removed
Counters.decrement(Counters.Counter) (flatcontract1.sol#1156-1164) is never used and should be removed
Counters.reset(Counters.Counter) (flatcontract1.sol#1166-1168) is never used and should be removed
ERC721._burn(uint256) (flatcontract1.sol#844-858) is never used and should be removed
ERC721._safeMint(address,uint256) (flatcontract1.sol#788-790) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (flatcontract1.sol#796-806) is never used and should be removed
ERC721URIStorage._burn(uint256) (flatcontract1.sol#1045-1051) is never used and should be removed
Strands._burn(uint256) (flatcontract1.sol#1220-1222) is never used and should be removed
Strings.toHexString(address) (flatcontract1.sol#524-528) is never used and should be removed
Strings.toHexString(uint256) (flatcontract1.sol#493-504) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.9 (flatcontract1.sol#4) 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
```



```
Low level call in Address.sendValue(address,uint256) (flatcontract1.sol#271-276):
        - (success) = recipient.call{value: amount}() (flatcontract1.sol#274)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (flatcontract1.sol#339-350):
        - (success,returndata) = target.call{value: value}(data) (flatcontract1.sol#348)
Low level call in Address.functionStaticCall(address,bytes,string) (flatcontract1.sol#368-377):
        - (success,returndata) = target.staticcall(data) (flatcontract1.sol#375)
Low level call in Address.functionDelegateCall(address,bytes,string) (flatcontract1.sol#395-404):
        - (success,returndata) = target.delegatecall(data) (flatcontract1.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Strands.setTransfer(bool)._enabletransfer (flatcontract1.sol#1185) is not in mixedCase
Parameter Strands.updateTokenURI(uint256,string)._tokenId (flatcontract1.sol#1237) is not in mixedCase
Parameter Strands.updateTokenURI(uint256,string)._tokenURI (flatcontract1.sol#1237) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Strands.contractOwner (flatcontract1.sol#1177) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (flatcontract1.sol#603-606)
name() should be declared external:
        - ERC721.name() (flatcontract1.sol#620-622)
symbol() should be declared external:
        - ERC721.symbol() (flatcontract1.sol#627-629)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (flatcontract1.sol#653-663)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (flatcontract1.sol#677-679)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (flatcontract1.sol#691-700)
        - Strands.transferFrom(address,address,uint256) (flatcontract1.sol#1190-1198)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (flatcontract1.sol#705-711)
        - Strands.safeTransferFrom(address,address,uint256) (flatcontract1.sol#1201-1208)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (flatcontract1.sol#1107-1109)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (flatcontract1.sol#1115-1118)
setTransfer(bool) should be declared external:
        - Strands.setTransfer(bool) (flatcontract1.sol#1185-1187)
mint(address) should be declared external:
        - Strands.mint(address) (flatcontract1.sol#1212-1218)
updateBaseMetadataApi(string) should be declared external:
        - Strands.updateBaseMetadataApi(string) (flatcontract1.sol#1231-1234)
updateTokenURI(uint256,string) should be declared external:
        - Strands.updateTokenURI(uint256,string) (flatcontract1.sol#1237-1239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
flatcontract1.sol analyzed (13 contracts with 78 detectors), 52 result(s) found
```

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

## Closing Summary

In this report, we have considered the security of the eDao contract. We performed our audit according to the procedure described above.

Some issues of High,Medium, low and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

## Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the eDao Platform. 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 eDao Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

## About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

| | | |
|---|---|---|
| **600+** | **$15B** | **600K** |
| Audits Completed | Secured | Lines of Code Audited |

## Follow Our Journey

# Audit Report
September, 2022

For

EDAO

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com