BLOCKHAT
SECURITY

Mimic Shhans - Official

Smart Contract Security Audit

Prepared by BlockHat

April 17th, 2023 - April 19th, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	Mimic Shhans
Version	0.1
Classification	Public

Scope

The Mimic Shhans - Official Contract in the Mimic Shhans - Official Repository

Link	Address
https://etherscan.io/address/0xF75FD01D2262b07D92dcA7f19bD6A3457060d7db#code	0xF75FD01D2262b07D92dcA7f19bD6A3457060d7db

Files	MD5 Hash
/MimicShhans.sol	5808a8e8d2dbe1e33dc075358fbaff53

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

Mimic Shhans - Official engaged BlockHat to conduct a security assessment on the Mimic Shhans - Official beginning on April 17th, 2023 and ending April 19th, 2023. 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 Mimic Shhans - Official

MIMIC SHHANS is a brand led and shaped by the community based on CC0. It is a new kind of brand made together by everyone around the world. Artist SHHAN will continue to expand on the Shhan Universe and develop the guide for Mimic Shhans brand identity.

Issuer	Mimic Shhans
Website	`https://mimicshhans.com/`
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.

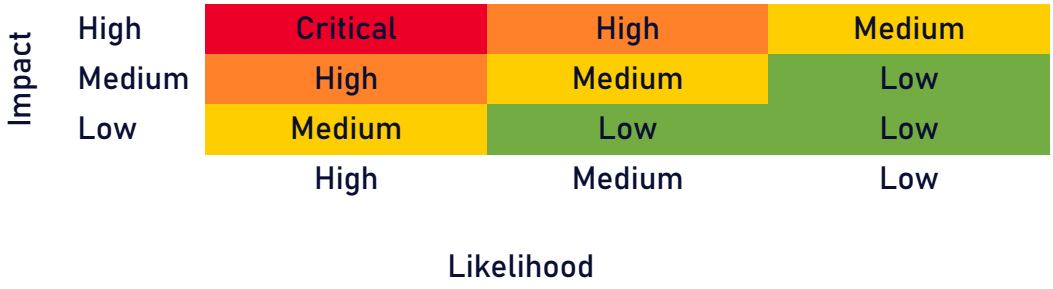
1.2.1 Risk Methodology

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 Mimic Shhans - Official 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 , 5 medium-severity, 1 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Unnecessary _mintAmount Parameter in mint Function	MEDIUM	Not fixed
Users Unable to Mint Tokens	MEDIUM	Not fixed
No Mechanism to Retrieve Excess Payments	MEDIUM	Not fixed
Missing Input Validation in setRoyaltyInfo Function	MEDIUM	Not fixed
Unrestricted setMaxMintAmountPerTx Function	MEDIUM	Not fixed
No Input Validation for setUriPrefix and setUriSuffix Functions	LOW	Not fixed

3 Finding Details

A MimicShhans.sol

A.1 Unnecessary _mintAmount Parameter in mint Function [MEDIUM]

Description:

The mint function has an unnecessary _mintAmount parameter, as it always mints only one token.

Code:

Listing 1: MimicShhans.sol

```
1467    function mint(uint256 _mintAmount, uint256 tokenId) public payable
           ↪ onlyOwner mintCompliance(_mintAmount) {
1468      require(!paused, "The contract is paused!");
1469      if (msg.sender != owner()) {
1470        require(msg.value >= cost * _mintAmount);
1471      }

1473      _mintMS(msg.sender, tokenId);
1474    }
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Remove the _mintAmount parameter from the mint function and update the mintCompliance modifier call to use a hardcoded value of 1.

Status - Not fixed

A.2 Users Unable to Mint Tokens [MEDIUM]

Description:

The mint and bulkmint functions are restricted to onlyOwner, preventing users from minting tokens.

Code:

Listing 2: MimicShhans.sol

```
1467    function mint(uint256 _mintAmount, uint256 tokenId) public payable
           ↪ onlyOwner mintCompliance(_mintAmount) {
1468      require(!paused, "The contract is paused!");
1469      if (msg.sender != owner()) {
1470        require(msg.value >= cost * _mintAmount);
1471      }
1473      _mintMS(msg.sender, tokenId);
1474    }
```

Listing 3: MimicShhans.sol

```
1475    function bulkmint(uint256 _mintAmount, uint256 tokenId) public payable
           ↪  onlyOwner mintCompliance(_mintAmount){
1476      require(!paused, "The contract is paused!");
1477      if (msg.sender != owner()) {
1478        require(msg.value >= cost * _mintAmount);
1479      }
1480      for (uint256 i = tokenId; i < tokenId + _mintAmount; i++) {
1481        _mintMS(msg.sender, i);
1482      }
1483    }
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Remove the onlyOwner modifier from the mint and bulkmint functions to allow users to mint tokens.

Status - Not fixed

A.3 No Mechanism to Retrieve Excess Payments [MEDIUM]

Description:

The contract does not have a mechanism to return excess payments if a user sends more than the required amount for minting tokens.

Code:

Listing 4: MimicShhans.sol

```
1467    function mint(uint256 _mintAmount, uint256 tokenId) public payable
            ↪ onlyOwner mintCompliance(_mintAmount) {
1468      require(!paused, "The contract is paused!");
1469      if (msg.sender != owner()) {
1470        require(msg.value >= cost * _mintAmount);
1471      }

1473      _mintMS(msg.sender, tokenId);
1474    }
```

Listing 5: MimicShhans.sol

```
1475    function bulkmint(uint256 _mintAmount, uint256 tokenId) public payable
            ↪  onlyOwner mintCompliance(_mintAmount){
```

```
1476      require(!paused, "The contract is paused!");
1477      if (msg.sender != owner()) {
1478        require(msg.value >= cost * _mintAmount);
1479      }
1480      for (uint256 i = tokenId; i < tokenId + _mintAmount; i++) {
1481        _mintMS(msg.sender, i);
1482      }
1483    }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Implement a mechanism to refund excess payments to the sender if they send more than the required amount for minting tokens. This can be achieved by calculating the excess amount and transferring it back to the sender using the transfer function of the address type.

Status - Not fixed

A.4 Missing Input Validation in setRoyaltyInfo Function [MEDIUM]

Description:

The setRoyaltyInfo function does not validate the input for the _royaltyFeesInBips parameter, allowing values greater than 10000 (100

Code:

Listing 6: MimicShhans.sol

```
1450    constructor(uint96 _royaltyFeesInBips, string memory _contractURI)
          ↪ ERC721("MimicShhans", "MS") {
1451      setHiddenMetadataUri("https://ipfs.filebase.io/ipfs/
            ↪ QmX3v9fNcEFikN6Wc6FvFKqcWD3RxwNhWiX6qzR7Yx6gpK/hidden.json");
1452      setRoyaltyInfo(msg.sender, _royaltyFeesInBips);
1453      contractURI = _contractURI;

1455    }
```

Listing 7: MimicShhans.sol

```
1585      function setRoyaltyInfo(address _receiver, uint96 _royaltyFeesInBips
            ↪ ) public onlyOwner
1586      {
1587          _setDefaultRoyalty(_receiver,_royaltyFeesInBips);
1588      }
```

Risk Level:

Likelihood – 2
Impact - 5

Recommendation:

Add a require statement to validate that _royaltyFeesInBips is less than or equal to 10000.

Status - Not fixed

A.5 Unrestricted setMaxMintAmountPerTx Function [MEDIUM]

Description:

The setMaxMintAmountPerTx function allows the owner to set the maxMintAmountPerTx to any arbitrary value without any restrictions, which may lead to potential abuse or unintended behavior.

Code:

Listing 8: MimicShhans.sol

```
1541    function setMaxMintAmountPerTx(uint256 _maxMintAmountPerTx) public
            ↪ onlyOwner {
1542      maxMintAmountPerTx = _maxMintAmountPerTx;
1543    }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Add a require statement to validate that the _maxMintAmountPerTx is within an acceptable range, such as less than or equal to the maxSupply.

Status - Not fixed

A.6 No Input Validation for setUriPrefix and setUriSuffix Functions [LOW]

Description:

The setUriPrefix and setUriSuffix functions do not validate the input, allowing the owner to set an empty or invalid URI prefix or suffix, which may lead to unexpected results when generating token URIs.

Code:

Listing 9: MimicShhans.sol

```
1549    function setUriPrefix(string memory _uriPrefix) public onlyOwner {
1550      uriPrefix = _uriPrefix;
```

```
1551    }

1553    function setUriSuffix(string memory _uriSuffix) public onlyOwner {
1554      uriSuffix = _uriSuffix;
1555    }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Add a require statement in both the setUriPrefix and setUriSuffix functions to ensure that the provided URI prefix and suffix are non-empty and meet the desired format.

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:

```
ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ MimicShhans.sol#1264-1286) ignores return value by
    ↪ IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,
    ↪ data) (MimicShhans.sol#1271-1282)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

MimicShhans.walletOfOwner(address)._owner (MimicShhans.sol#1490) shadows
    ↪ :
        - Ownable._owner (MimicShhans.sol#178) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ MimicShhans.sol#1264-1286) has external calls inside a loop:
    ↪ IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,
    ↪ data) (MimicShhans.sol#1271-1282)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop
```

```
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ retval (MimicShhans.sol#1271)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (MimicShhans.sol#1264-1286)
    ↪ potentially used before declaration: retval == IERC721Receiver.
    ↪ onERC721Received.selector (MimicShhans.sol#1272)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (MimicShhans.sol#1273)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (MimicShhans.sol#1264-1286)
    ↪ potentially used before declaration: reason.length == 0 (
    ↪ MimicShhans.sol#1274)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (MimicShhans.sol#1273)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (MimicShhans.sol#1264-1286)
    ↪ potentially used before declaration: revert(uint256,uint256)(32 +
    ↪  reason,mload(uint256)(reason)) (MimicShhans.sol#1279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Address.verifyCallResult(bool,bytes,string) (MimicShhans.sol#444-464)
    ↪ uses assembly
        - INLINE ASM (MimicShhans.sol#456-459)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ MimicShhans.sol#1264-1286) uses assembly
        - INLINE ASM (MimicShhans.sol#1278-1280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

MimicShhans.tokenURI(uint256) (MimicShhans.sol#1515-1535) compares to a
    ↪ boolean constant:
        -revealed == false (MimicShhans.sol#1527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Different versions of Solidity are used:
```

```
        - Version used: ['>=0.7.0<0.9.0', '^0.8.0', '^0.8.1']
        - ^0.8.0 (MimicShhans.sol#11)
        - ^0.8.0 (MimicShhans.sol#57)
        - ^0.8.0 (MimicShhans.sol#135)
        - ^0.8.0 (MimicShhans.sol#162)
        - ^0.8.1 (MimicShhans.sol#247)
        - ^0.8.0 (MimicShhans.sol#472)
        - ^0.8.0 (MimicShhans.sol#502)
        - ^0.8.0 (MimicShhans.sol#530)
        - ^0.8.0 (MimicShhans.sol#557)
        - ^0.8.0 (MimicShhans.sol#588)
        - ^0.8.0 (MimicShhans.sol#701)
        - ^0.8.0 (MimicShhans.sol#846)
        - ^0.8.0 (MimicShhans.sol#875)
        - ^0.8.0 (MimicShhans.sol#1331)
        - ^0.8.0 (MimicShhans.sol#1395)
        - >=0.7.0<0.9.0 (MimicShhans.sol#1422)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
       ↪ #different-pragma-directives-are-used

    Address.functionCall(address,bytes) (MimicShhans.sol#328-330) is never
       ↪ used and should be removed
    Address.functionCall(address,bytes,string) (MimicShhans.sol#338-344) is
       ↪ never used and should be removed
    Address.functionCallWithValue(address,bytes,uint256) (MimicShhans.sol
       ↪ #357-363) is never used and should be removed
    Address.functionCallWithValue(address,bytes,uint256,string) (MimicShhans
       ↪ .sol#371-382) is never used and should be removed
    Address.functionDelegateCall(address,bytes) (MimicShhans.sol#417-419) is
       ↪  never used and should be removed
    Address.functionDelegateCall(address,bytes,string) (MimicShhans.sol
       ↪ #427-436) is never used and should be removed
    Address.functionStaticCall(address,bytes) (MimicShhans.sol#390-392) is
       ↪ never used and should be removed
```

```
Address.functionStaticCall(address,bytes,string) (MimicShhans.sol
   ↪ #400-409) is never used and should be removed
Address.sendValue(address,uint256) (MimicShhans.sol#303-308) is never
   ↪ used and should be removed
Address.verifyCallResult(bool,bytes,string) (MimicShhans.sol#444-464) is
   ↪  never used and should be removed
Context._msgData() (MimicShhans.sol#152-154) is never used and should be
   ↪  removed
Counters.decrement(Counters.Counter) (MimicShhans.sol#39-45) is never
   ↪ used and should be removed
Counters.reset(Counters.Counter) (MimicShhans.sol#47-49) is never used
   ↪ and should be removed
ERC2981._deleteDefaultRoyalty() (MimicShhans.sol#665-667) is never used
   ↪ and should be removed
ERC2981._resetTokenRoyalty(uint256) (MimicShhans.sol#691-693) is never
   ↪ used and should be removed
ERC2981._setTokenRoyalty(uint256,address,uint96) (MimicShhans.sol
   ↪ #677-686) is never used and should be removed
ERC721._baseURI() (MimicShhans.sol#975-977) is never used and should be
   ↪ removed
ERC721URIStorage._setTokenURI(uint256,string) (MimicShhans.sol
   ↪ #1371-1374) is never used and should be removed
Strings.toHexString(address) (MimicShhans.sol#125-127) is never used and
   ↪  should be removed
Strings.toHexString(uint256) (MimicShhans.sol#94-105) is never used and
   ↪ should be removed
Strings.toHexString(uint256,uint256) (MimicShhans.sol#110-120) is never
   ↪ used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.0 (MimicShhans.sol#11) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#57) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#135) allows old versions
```

```
Pragma version^0.8.0 (MimicShhans.sol#162) allows old versions
Pragma version^0.8.1 (MimicShhans.sol#247) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#472) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#502) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#530) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#557) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#588) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#701) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#846) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#875) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#1331) allows old versions
Pragma version^0.8.0 (MimicShhans.sol#1395) allows old versions
Pragma version>=0.7.0<0.9.0 (MimicShhans.sol#1422) is too complex
solc-0.8.17 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) (MimicShhans.sol
    ↪ #303-308):
        - (success) = recipient.call{value: amount}() (MimicShhans.sol
            ↪ #306)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (MimicShhans.sol#371-382):
        - (success,returndata) = target.call{value: value}(data) (
            ↪ MimicShhans.sol#380)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ MimicShhans.sol#400-409):
        - (success,returndata) = target.staticcall(data) (MimicShhans.sol
            ↪ #407)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ MimicShhans.sol#427-436):
        - (success,returndata) = target.delegatecall(data) (MimicShhans.
            ↪ sol#434)
Low level call in MimicShhans.withdraw() (MimicShhans.sol#1561-1564):
```

```
        - (os) = address(owner()).call{value: address(this).balance}() (
            ↪ MimicShhans.sol#1562)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter ERC2981.royaltyInfo(uint256,uint256)._tokenId (MimicShhans.sol
    ↪ #626) is not in mixedCase
Parameter ERC2981.royaltyInfo(uint256,uint256)._salePrice (MimicShhans.
    ↪ sol#626) is not in mixedCase
Parameter MimicShhans.mint(uint256,uint256)._mintAmount (MimicShhans.sol
    ↪ #1467) is not in mixedCase
Parameter MimicShhans.bulkmint(uint256,uint256)._mintAmount (MimicShhans
    ↪ .sol#1475) is not in mixedCase
Parameter MimicShhans.walletOfOwner(address)._owner (MimicShhans.sol
    ↪ #1490) is not in mixedCase
Parameter MimicShhans.tokenURI(uint256)._tokenId (MimicShhans.sol#1515)
    ↪ is not in mixedCase
Parameter MimicShhans.setRevealed(bool)._state (MimicShhans.sol#1537) is
    ↪  not in mixedCase
Parameter MimicShhans.setMaxMintAmountPerTx(uint256)._maxMintAmountPerTx
    ↪  (MimicShhans.sol#1541) is not in mixedCase
Parameter MimicShhans.setHiddenMetadataUri(string)._hiddenMetadataUri (
    ↪ MimicShhans.sol#1545) is not in mixedCase
Parameter MimicShhans.setUriPrefix(string)._uriPrefix (MimicShhans.sol
    ↪ #1549) is not in mixedCase
Parameter MimicShhans.setUriSuffix(string)._uriSuffix (MimicShhans.sol
    ↪ #1553) is not in mixedCase
Parameter MimicShhans.setPaused(bool)._state (MimicShhans.sol#1557) is
    ↪ not in mixedCase
Parameter MimicShhans.setRoyaltyInfo(address,uint96)._receiver (
    ↪ MimicShhans.sol#1585) is not in mixedCase
Parameter MimicShhans.setRoyaltyInfo(address,uint96)._royaltyFeesInBips
    ↪ (MimicShhans.sol#1585) is not in mixedCase
```

```
Parameter MimicShhans.setContractURI(string)._contractURI (MimicShhans.
    ↪ sol#1590) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

MimicShhans.cost (MimicShhans.sol#1440) should be constant
MimicShhans.maxSupply (MimicShhans.sol#1441) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MimicShhans.sol#218-220)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MimicShhans.sol#226-229)
royaltyInfo(uint256,uint256) should be declared external:
        - ERC2981.royaltyInfo(uint256,uint256) (MimicShhans.sol#626-636)
name() should be declared external:
        - ERC721.name() (MimicShhans.sol#949-951)
symbol() should be declared external:
        - ERC721.symbol() (MimicShhans.sol#956-958)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (MimicShhans.sol#982-992)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (MimicShhans.sol
            ↪ #1006-1008)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (MimicShhans.sol
            ↪ #1020-1029)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (MimicShhans.
            ↪ sol#1034-1040)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (MimicShhans.sol#1411-1415)
totalSupply() should be declared external:
```

```
        - MimicShhans.totalSupply() (MimicShhans.sol#1463-1465)
mint(uint256,uint256) should be declared external:
        - MimicShhans.mint(uint256,uint256) (MimicShhans.sol#1467-1474)
bulkmint(uint256,uint256) should be declared external:
        - MimicShhans.bulkmint(uint256,uint256) (MimicShhans.sol
          ↪ #1475-1483)
walletOfOwner(address) should be declared external:
        - MimicShhans.walletOfOwner(address) (MimicShhans.sol#1490-1513)
setRevealed(bool) should be declared external:
        - MimicShhans.setRevealed(bool) (MimicShhans.sol#1537-1539)
setMaxMintAmountPerTx(uint256) should be declared external:
        - MimicShhans.setMaxMintAmountPerTx(uint256) (MimicShhans.sol
          ↪ #1541-1543)
setUriPrefix(string) should be declared external:
        - MimicShhans.setUriPrefix(string) (MimicShhans.sol#1549-1551)
setUriSuffix(string) should be declared external:
        - MimicShhans.setUriSuffix(string) (MimicShhans.sol#1553-1555)
setPaused(bool) should be declared external:
        - MimicShhans.setPaused(bool) (MimicShhans.sol#1557-1559)
withdraw() should be declared external:
        - MimicShhans.withdraw() (MimicShhans.sol#1561-1564)
setContractURI(string) should be declared external:
        - MimicShhans.setContractURI(string) (MimicShhans.sol#1590-1592)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #public-function-that-could-be-declared-external
MimicShhans.sol analyzed (16 contracts with 78 detectors), 91 result(s)
   ↪ found
```

Conclusion:

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

5 Conclusion

We examined the design and implementation of Mimic Shhans - Official in this audit and found several issues of various severities. We advise Mimic Shhans team to implement the recommendations contained in all 6 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 Contract Audit, contact us at contact@blockhat.io