Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Xandao Pixels
Platform: Etherscan
Language: Solidity
Date: October 11th, 2023

Table of contents

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Club Hush team to perform the Security audit of the Xandao Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on October 11th, 2023.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- Xandao Pixels is an NFT project defined by a 1-36-char tokenid, representing a single pixel in six colors.
- Each piece is on an oblique 6x6 canvas, rendered on a chain as a svg.
- The creator is permanently recorded on the chain, with a self-declared description.
- The aesthetic traits are derived from the token id, and the price to mint depends on the token id length.
- The price is a geometric progression, starting at 0.006 Eth for a 36-char token id and 6 Eth for a single char tokenid.
- The Xandao Pixels contract inherits ERC721, ERC721Burnable, Ownable2Step, Counters, Base64, Strings, SafeMath standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Xandao Pixels Smart Contract
Platform	**Etherscan / Solidity**
File	PixelsV1.sol
MD5 hash code	DE1A35DDCBEEBEE0AD04D03383D501F1
Updated MD5 hash code	7DDEABBC1C06AE805FCD9F82340BEA23
Audit Date	October 11th, 2023
Revised Audit Date	October 12th, 2023

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Ownership Control: ● Withdraw amount. ● Current owner can transfer the ownership. ● Owner can renounce ownership.	**YES, This is valid.**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. **Also, these contracts contain owner control, which does not make them fully decentralized.**

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 3 low and 0 very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Xandao Pixels are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Xandao Pixels.

The Xandao Pixels team has not provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not well** commented on in the smart contracts. Ethereum's NatSpec commenting style is used.

Documentation

We were given a Xandao Pixels smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented but the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setMintPrice	write	Removed	-
3	_checkTokenId	internal	Passed	No Issue
4	getAllTokenIds	read	Passed	No Issue
5	upgradePixels	write	Passed	No Issue
6	getMintPrice	read	Passed	No Issue
7	mintPixels	write	Function input parameters lack of check	Refer Audit Findings
8	tokenURIByTokenId	read	Passed	No Issue
9	tokenURI	read	Passed	No Issue
10	tokenURIByXN	read	Passed	No Issue
11	creatorOf	read	Passed	No Issue
12	isAvailable	read	Passed	No Issue
13	getBalance	read	Passed	No Issue
14	withdraw	write	Owner can drain fund	Refer Audit Findings
15	pendingOwner	read	Passed	No Issue
16	transferOwnership	write	access only Owner	No Issue
17	_transferOwnership	internal	Passed	No Issue
18	acceptOwnership	write	Passed	No Issue
19	onlyOwner	modifier	Passed	No Issue
20	owner	read	Passed	No Issue
21	_checkOwner	internal	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue
24	_transferOwnership	internal	Passed	No Issue
25	supportsInterface	read	Passed	No Issue
26	balanceOf	read	Passed	No Issue
27	ownerOf	read	Passed	No Issue
28	name	read	Passed	No Issue
29	symbol	read	Passed	No Issue
30	tokenURI	read	Passed	No Issue
31	_baseURI	internal	Passed	No Issue
32	approve	write	Passed	No Issue
33	getApproved	read	Passed	No Issue
34	setApprovalForAll	write	Passed	No Issue
35	isApprovedForAll	read	Passed	No Issue
36	transferFrom	write	Passed	No Issue
37	safeTransferFrom	write	Passed	No Issue
38	safeTransferFrom	write	Passed	No Issue
39	_ownerOf	internal	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

40	_getApproved	internal	Passed	No Issue
41	_isAuthorized	internal	Passed	No Issue
42	_checkAuthorized	internal	Passed	No Issue
43	_increaseBalance	internal	Passed	No Issue
44	_update	internal	Passed	No Issue
45	_mint	internal	Passed	No Issue
46	_safeMint	internal	Passed	No Issue
47	_safeMint	internal	Passed	No Issue
48	_burn	internal	Passed	No Issue
49	_transfer	internal	Passed	No Issue
50	_safeTransfer	internal	Passed	No Issue
51	_safeTransfer	internal	Passed	No Issue
52	_approve	internal	Passed	No Issue
53	_approve	internal	Passed	No Issue
54	_setApprovalForAll	internal	Passed	No Issue
55	_requireOwned	internal	Passed	No Issue
56	_checkOnERC721Received	write	Passed	No Issue
57	generateSVG	read	Division before multiplication	Refer Audit Findings
58	generateTokenURI	read	Passed	No Issue
59	handlePadding	write	Passed	No Issue
60	getDigitCounts	write	Passed	No Issue
61	getColorHSL	write	Passed	No Issue
62	averageHSL	read	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Division before multiplication: **PixelsMetadataUtils.sol**

```
function generateSVG(uint256 tokenId, string memory gridSize) public view returns (string memory) {

    string memory svgContent = getBasePart1(gridSize);
    for (uint i = 0; i < 36; i++) {
        uint spiralIndex = spiralKey[i] - 1;
        uint colorIndex = (tokenId / 10 ** (36 - i - 1)) % 10;

        string memory x = (spiralIndex % 6).toString();
        string memory y = (spiralIndex / 6).toString();

        string memory rectangle = string(
            abi.encodePacked(
```

The generateSVG() function to calculate colorIndex, first performing a division operation tokenId / 10 ** (36 - i - 1) and then taking the modulo.

Resolution: This can be optimized by performing the modulo operation first, and then the division.

(2) Function input parameters lack of check: **PixelsV1.sol**

Some functions require validation before execution.

Functions are:

- mintPixels() - creator

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Resolution: We suggest using validation like for numerical variables that should be greater than 0 and for address type check variables that are not address(0). For percentage type variables, values should have some range like minimum 0 and maximum 100.

(3) Owner can drain fund: **PixelsV1.sol**
Owner can drain all the balance of the contract.

Resolution: We suggest confirming withdrawal functionality before moving Mainnet, If this is a part of the plan then disregard this issue.

Very Low / Informational / Best practices:

No very low severity vulnerabilities were found.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

PixelsV1.sol

- withdraw: withdraw by the owner.

Ownable2Step.sol

- transferOwnership: Current owner can transfer ownership of the contract to a new account.

Ownable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transfer ownership: The current owner can transfer ownership of the contract to a new account.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 3 low issues in the smart contracts. but those are not critical. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

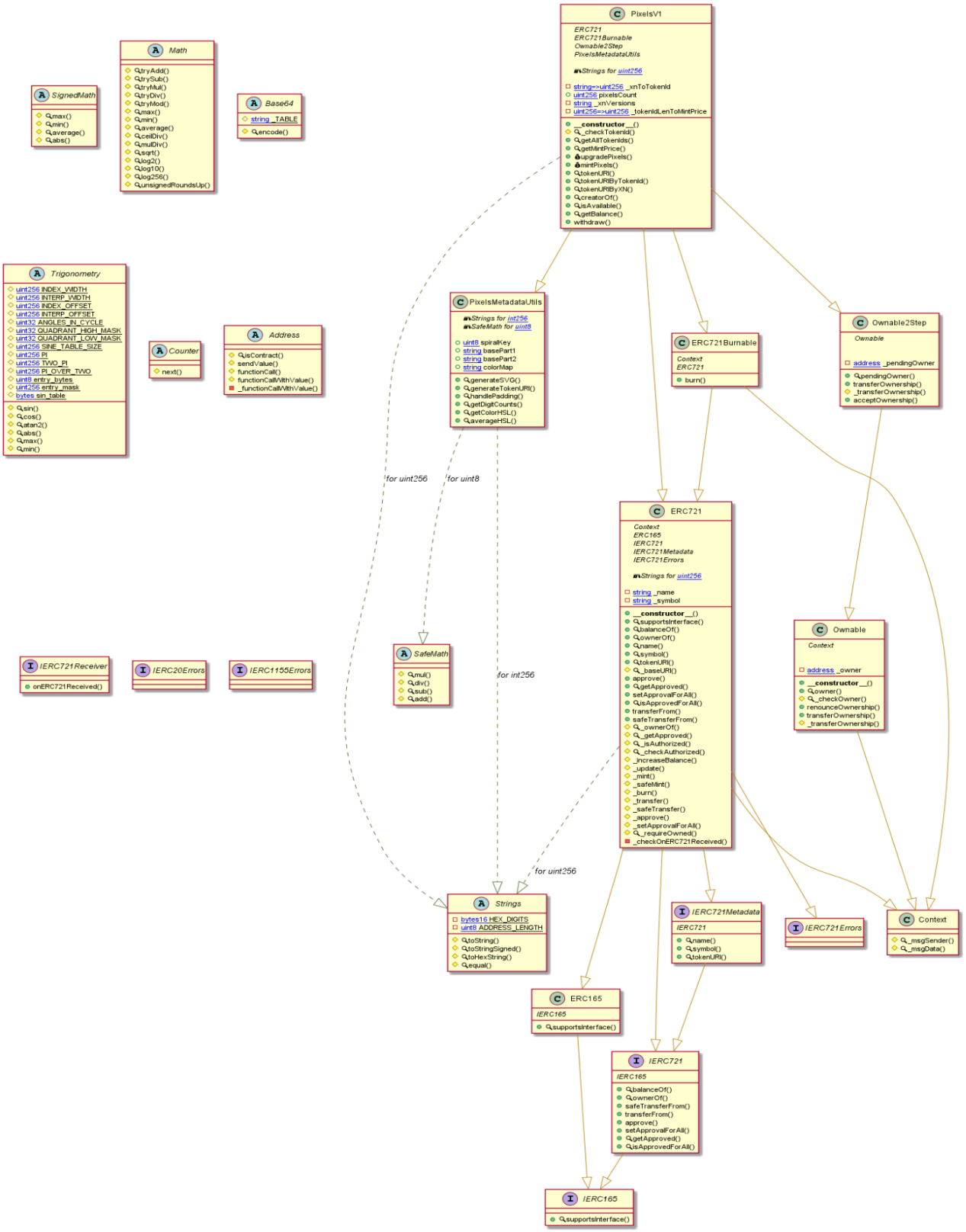
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Appendix

Code Flow Diagram - Xandao Pixels

PixelsV1 Diagram



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> PixelsV1.sol

```
PixelsV1.constructor(string,string,uint256).name (PixelsV1.sol#1929) shadows:
        - ERC721.name() (PixelsV1.sol#1384-1386) (function)
        - IERC721Metadata.name() (PixelsV1.sol#1151) (function)
PixelsV1.constructor(string,string,uint256).symbol (PixelsV1.sol#1930) shadows:
        - ERC721.symbol() (PixelsV1.sol#1391-1393) (function)
        - IERC721Metadata.symbol() (PixelsV1.sol#1153) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

PixelsV1.setMintPrice(uint256) (PixelsV1.sol#1937-1939) should emit an event for:
        - _mintPrice = mintPrice (PixelsV1.sol#1938)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Ownable2Step.transferOwnership(address).newOwner (PixelsV1.sol#1893) lacks a zero-check on :
                - _pendingOwner = newOwner (PixelsV1.sol#1894)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (PixelsV1.sol#1777)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (PixelsV1.sol#1775-1792) potentially used before declaration: retval != IERC721Receiver.onE
RC721Received.selector (PixelsV1.sol#1778)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (PixelsV1.sol#1781)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (PixelsV1.sol#1775-1792) potentially used before declaration: reason.length == 0 (PixelsV1.
sol#1782)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (PixelsV1.sol#1781)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (PixelsV1.sol#1775-1792) potentially used before declaration: revert(uint256,uint256)(32 +
reason,mload(uint256)(reason)) (PixelsV1.sol#1787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Math.mulDiv(uint256,uint256,uint256) (PixelsV1.sol#210-289) uses assembly
        - INLINE ASM (PixelsV1.sol#217-220)
        - INLINE ASM (PixelsV1.sol#241-248)
        - INLINE ASM (PixelsV1.sol#254-263)
Strings.toString(uint256) (PixelsV1.sol#516-536) uses assembly
        - INLINE ASM (PixelsV1.sol#522-524)
        - INLINE ASM (PixelsV1.sol#528-530)
Base64.encode(bytes) (PixelsV1.sol#597-668) uses assembly
        - INLINE ASM (PixelsV1.sol#616-665)
Trigonometry.sin(uint256) (PixelsV1.sol#711-771) uses assembly
        - INLINE ASM (PixelsV1.sol#743-746)
Address.isContract(address) (PixelsV1.sol#1026-1033) uses assembly
        - INLINE ASM (PixelsV1.sol#1029-1031)
Address._functionCallWithValue(address,bytes,uint256,string) (PixelsV1.sol#1072-1094) uses assembly
        - INLINE ASM (PixelsV1.sol#1086-1089)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (PixelsV1.sol#1775-1792) uses assembly
        - INLINE ASM (PixelsV1.sol#1786-1788)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (PixelsV1.sol#1072-1094) is never used and should be removed
Address.functionCall(address,bytes) (PixelsV1.sol#1042-1044) is never used and should be removed
Address.functionCall(address,bytes,string) (PixelsV1.sol#1046-1052) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PixelsV1.sol#1054-1060) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (PixelsV1.sol#1062-1070) is never used and should be removed
Address.isContract(address) (PixelsV1.sol#1026-1033) is never used and should be removed
Address.sendValue(address,uint256) (PixelsV1.sol#1035-1040) is never used and should be removed
Base64.encode(bytes) (PixelsV1.sol#597-668) is never used and should be removed
Context._msgData() (PixelsV1.sol#1316-1319) is never used and should be removed
Counter.next(Counter.Counter) (PixelsV1.sol#1016-1022) is never used and should be removed
ERC721._baseURI() (PixelsV1.sol#1410-1412) is never used and should be removed
ERC721._burn(uint256) (PixelsV1.sol#1638-1643) is never used and should be removed
ERC721._increaseBalance(address,uint128) (PixelsV1.sol#1534-1538) is never used and should be removed
ERC721._safeTransfer(address,address,uint256) (PixelsV1.sol#1687-1689) is never used and should be removed
ERC721._safeTransfer(address,address,uint256,bytes) (PixelsV1.sol#1695-1698) is never used and should be removed
ERC721._transfer(address,address,uint256) (PixelsV1.sol#1656-1666) is never used and should be removed
Math.average(uint256,uint256) (PixelsV1.sol#183-186) is never used and should be removed
```

```
Math.ceilDiv(uint256,uint256) (PixelsV1.sol#194-202) is never used and should be removed
Math.log10(uint256,Math.Rounding) (PixelsV1.sol#446-451) is never used and should be removed
Math.log2(uint256) (PixelsV1.sol#355-391) is never used and should be removed
Math.log2(uint256,Math.Rounding) (PixelsV1.sol#397-402) is never used and should be removed
Math.log256(uint256) (PixelsV1.sol#459-483) is never used and should be removed
Math.log256(uint256,Math.Rounding) (PixelsV1.sol#489-494) is never used and should be removed
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
Math.max(uint256,uint256) (PixelsV1.sol#168-170) is never used and should be removed
Math.min(uint256,uint256) (PixelsV1.sol#175-177) is never used and should be removed
Math.mulDiv(uint256,uint256,uint256) (PixelsV1.sol#210-289) is never used and should be removed
Math.mulDiv(uint256,uint256,uint256,Math.Rounding) (PixelsV1.sol#294-300) is never used and should be removed
Math.sqrt(uint256) (PixelsV1.sol#308-339) is never used and should be removed
Math.sqrt(uint256,Math.Rounding) (PixelsV1.sol#344-349) is never used and should be removed
Math.tryAdd(uint256,uint256) (PixelsV1.sol#112-118) is never used and should be removed
Math.tryDiv(uint256,uint256) (PixelsV1.sol#148-153) is never used and should be removed
Math.tryMod(uint256,uint256) (PixelsV1.sol#158-163) is never used and should be removed
Math.tryMul(uint256,uint256) (PixelsV1.sol#133-143) is never used and should be removed
Math.trySub(uint256,uint256) (PixelsV1.sol#123-128) is never used and should be removed
Math.unsignedRoundsUp(Math.Rounding) (PixelsV1.sol#499-501) is never used and should be removed
SafeMath.add(uint256,uint256) (PixelsV1.sol#53-57) is never used and should be removed
SafeMath.div(uint256,uint256) (PixelsV1.sol#35-40) is never used and should be removed
SafeMath.mul(uint256,uint256) (PixelsV1.sol#23-30) is never used and should be removed
SafeMath.sub(uint256,uint256) (PixelsV1.sol#45-48) is never used and should be removed
SignedMath.abs(int256) (PixelsV1.sol#88-93) is never used and should be removed
SignedMath.average(int256,int256) (PixelsV1.sol#79-83) is never used and should be removed
SignedMath.max(int256,int256) (PixelsV1.sol#64-66) is never used and should be removed
SignedMath.min(int256,int256) (PixelsV1.sol#71-73) is never used and should be removed
Strings.equal(string,string) (PixelsV1.sol#583-585) is never used and should be removed
Strings.toHexString(address) (PixelsV1.sol#576-578) is never used and should be removed
Strings.toHexString(uint256) (PixelsV1.sol#548-552) is never used and should be removed
Strings.toHexString(uint256,uint256) (PixelsV1.sol#557-570) is never used and should be removed
Strings.toStringSigned(int256) (PixelsV1.sol#541-543) is never used and should be removed
Trigonometry.max(int256,int256) (PixelsV1.sol#811-813) is never used and should be removed
Trigonometry.min(int256,int256) (PixelsV1.sol#815-817) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.9 (PixelsV1.sol#5) allows old versions
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) (PixelsV1.sol#1035-1040):
        - (success) = recipient.call{value: amount}() (PixelsV1.sol#1038)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PixelsV1.sol#1072-1094):
        - (success,returndata) = target.call{value: weiValue}(data) (PixelsV1.sol#1080)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Trigonometry.sin(uint256)._angle (PixelsV1.sol#711) is not in mixedCase
Parameter Trigonometry.cos(uint256)._angle (PixelsV1.sol#781) is not in mixedCase
Constant Trigonometry.entry_bytes (PixelsV1.sol#696) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Trigonometry.entry_mask (PixelsV1.sol#697) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Trigonometry.sin_table (PixelsV1.sol#698-699) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PixelsV1._mintPrice (PixelsV1.sol#1922) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (PixelsV1.sol#1317)" inContext (PixelsV1.sol#1311-1320)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Trigonometry.slitherConstructorConstantVariables() (PixelsV1.sol#673-818) uses literals with too many digits:
        - sin_table = 0x0000000000c90f8801921d20025b26d703242abf03ed26e604b6195d057f00350647d97c0710a34507d95b9e08a2009a096a904
90a3308bc0afb68050bc3ac350c8bd35e0d53db920e1bc2e40ee387660fab272b1072a0481139f0cf120116d512c8106e138edbb1145576b1151bdf8515e214
4416a81305176dd9de183366e818f8b83c19bdcbf31a82a0251b4732ef1c0b826a1ccf8cb31d934fe51e56ca1e1f19f97b1fdcdc1b209f701c2161b39f2223a
4c522e541af23a6887e2467775725280c5d25e845b626a8218527679df42826b92828e5714a29a3c4852a61b1012b1f34eb2bdc4e6f2c98fbba2d553afb2e11
0a622ecc681e2f8752623041c76030fbc54d31b54a5d326e54c73326e2c233def2873496824f354d905636041ad936ba2013376f9e46382493b038d8fe93398
cdd323a402dd13af2eeb73ba51e293c56ba703d07c1d53db832a53e680b2c3f1749b73fc5ec974073f21d4121589a41ce1e64427a41d04325c13543d09aec44
7acd50452456bc45cd358f46756827471cece647c3c22e4869e664490f57ee49b415334a581c9d4afb6c974b9e038f4c3fdff34ce100344d8162c34e2106174
ebfe8a44f5e08e24ffb654c5097fc5e5133cc9451ced46e5269126e53028517539b2aef5433027d54ca0a4a556040e255f5a4d2568a34a9571deef957b0d255
5842dd5458d40e8c5964649759f3de125a8279995b1035ce5b9d11535c290acc5cb420df5d3e52365dc79d7b5e50015d5ed77c895f5e0db25fe3b38d60686cc
e60ec382f616f146b61f1003e6271fa6862f201ac637114cc63ef328f646c59bf64e889256563bf9165ddfbd266573cbb66cf811f6746c7d767bd0fbc683257
aa68a69e806919e31f698c246b69fd614a6a6d98a36adcc9646b4af2786bb812d06c24295f6c8f351b6cf934fb6d6227f96dca0d146e30e3496e96a99c6efb5
f116f5f02b16fc193847023109970837 8fe70e2cbc571410804719e2cd171fa394872552c8472af05a67307c3cf735f662573b5ebd0740b53fa745f9dd074b2
c8837504d3447555bd4b75a585ce75f42c0a7641af3c768e0ea576d9498877235f2c776c4eda77b417df77fab9887840332878848 41378c7aba17909a92c794
a7c11798a23b079c89f6d7a05eeac7a4210d87a7d055a7ab6cba37aef63237b26cb4e7b5d039d7b920b887bc5e28f7bf8882f7c29fbed7c5a3d4f7c894bdd7c
b727237ce3ceb17d0f42177d3980eb7d628ac57d8a5f3f7db0fdf77dd6668e7dfa98a77e1d93e97e3f57fe7e5fe4927e7f39567e9d55fb7eba3a387ed5e5c57
ef0585f7f0991c37f2191b37f3857f57f4de4507f62368e7f754e7f7f872bf27f97cebc7fa736b37fb563b27fc255957fce0c3d7fd8878d7fe1c76a7fe9cbbf
7ff094777ff621817ffa72d07ffd88597fff62157fffffff (PixelsV1.sol#698-699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
PixelsMetadataUtils.basePart1 (PixelsV1.sol#830-831) should be constant
PixelsMetadataUtils.basePart2 (PixelsV1.sol#832) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
PixelsV1.sol analyzed (22 contracts with 84 detectors), 95 result(s) found
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Solidity Static Analysis

PixelsV1.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 101:12:

Gas costs:

Gas requirement of function PixelsV1.withdraw is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 154:97:

Constant/View/Pure functions:

PixelsV1.tokenURIByXN(uint256) : Is constant but potentially should not be.
Note: Modifiers are currently not considered by this static analysis.

more

Pos: 135:5:

Similar variable names:

Trigonometry.sin(uint256) : Variables have very similar names "x1_2" and "x2".
Note: Modifiers are currently not considered by this static analysis.
Pos: 112:26:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 152:16:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 154:158:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 36:31:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Solhint Linter

PixelsV1.sol

```
Compiler version ^0.8.9 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:28
global import of path @openzeppelin/contracts/token/ERC721/ERC721.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:30
global import of path @openzeppelin/contracts/access/Ownable2Step.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:31
global import of path @openzeppelin/contracts/utils/Counters.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:32
global import of path ./PixelsMetadataUtils.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:33
global import of path ./PixelsTypesV1.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:34
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:45
Error message for require is too long
Pos: 9:68
Error message for require is too long
Pos: 9:74
Error message for require is too long
Pos: 13:104
Error message for require is too long
Pos: 9:129
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io