Cyberscope

Audit Report

Xandao

December 2023

Repository https://github.com/xandao-xo/xandao-smart-contracts

Commit 7abd4fbcf77e4a8efd44394cf1ebcfd142c584d8

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Overview 4
theXanDAO Contract 4
XandaoMeta Contract 4
Functionality 4
Findings Breakdown 6
Diagnostics 7
DCGI - Digit Count Gas Inefficiency 8
Description 8
Recommendation 8
MU - Modifiers Usage 9
Description 9
Recommendation 9
AISRC - Ascending If Statements Redundant Condition 10
Description 10
Recommendation 10
CCO - Color Code Optimization 11
Description 11
Recommendation 11
ECXH - Edge Case XN Handling 12
Description 12
Recommendation 13
MEM - Misleading Error Messages 14
Description 14
Recommendation 14
Functions Analysis 15
Inheritance Graph 17
Flow Graph 18
Summary 19
Disclaimer 20
About Cyberscope 21

Review

Repository	https://github.com/xandao-xo/xandao-smart-contracts
Commit	7abd4fbcf77e4a8efd44394cf1ebcfd142c584d8

Audit Updates

Initial Audit	30 Nov 2023
Corrected Phase 2	16 Dec 2023
Corrected Phase 3	21 Dec 2023

Source Files

Filename	SHA256
XandaoTypes.sol	2e64dffb019bc73070e1b904d31b1e544636b80981545ed0cbb02f9508ae9232
XandaoMeta.sol	c19cb3c9c75d752a7513ae2f17499b611f5d82e1ba9a84c7563d227afd048df2
Trigonometry.sol	2931db4219fa0676e5a11716e3e6ee6043e585d97e321feb611b28c25b50684b
TheXanDAO.sol	1c2be8c0c5badadbd3237b0991d5e99df8e738d1237c730a0da107adea14ab7d
Strings.sol	d81a4a55da7145b070af1100e65d0437bee08606024266d5c39f116c1fd6ca47
SignedMath.sol	420a5a5d8d94611a04b39d6cf5f02492552ed4257ea82aba3c765b1ad52f77f6
Ownable2Step.sol	3e3bdb084bc14ade54e8259e710287956a7dbf2b2b4ad1e4cd8899d2293c7241

Ownable.sol	33422e7771fefe5fbfe8934837515097119d82a50eda0e49b38e4d6a64a1c25d
Math.sol	85a2caf3bd06579fb55236398c1321e15fd524a8fe140dff748c0f73d7a52345
IERC721Receiver.sol	77f0f7340c2da6bb9edbc90ab6e7d3eb8e2ae18194791b827a3e8c0b11a09b43
IERC721Metadata.sol	fc0ee82753767d429965338d8e79830f2350908ed6c51b538f7b0f8b15b4224c
IERC721.sol	7a265b34318bc9845f06fb9a80f1d7ac32155db0ec0357ed8bf08f09743011ed
IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
ERC721Burnable.sol	c0f4a5e7df8396227b907fe60c6ed8ed472557d833e99993d4b1ad7cb8a4fdac
ERC721.sol	49704a47f37fc5ce0b85bc3a26d161fd0a6f6c9768c35da86f67bf331672ad54
ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
Counters.sol	2fdcb1343e5621385b62e57b5c7775607c272122b6f2dc77da8f84828aa40cd0
Context.sol	b2cfee351bcafd0f8f27c72d76c054df9b571b62cfac4781ed12c86354e2a56c
Base64.sol	9fbd7a4462f54bbb6b0bd03231738e5f081a092e9a8fd789fb4d1aeca3758aec
Address.sol	8b85a2463eda119c2f42c34fa3d942b61aee65df381f48ed436fe8edb3a7d602

Overview

This document presents the overview of the smart contract audit conducted for the "theXanDAO" and "XandaoMeta" contracts. The purpose of this audit is to identify and address security vulnerabilities, provide recommendations for code improvements, and ensure the robustness of the codebase.

The audited contracts demonstrate a solid foundation with proper usage of ERC721 and OpenZeppelin libraries. Recommendations have been provided to enhance security and functionality.

theXanDAO Contract

Manages the main functionality of the TheXanDAO project, including minting and upgrading of NFTs.

Implements an ERC721 token for the TheXanDAO project. Supports the minting and upgrading of NFTs based on a 36-character token ID. Handles ownership, burning, and token URI generation. Includes a price mechanism based on the length of the token ID.

XandaoMeta Contract

Provides utility functions for generating SVG and token URI metadata. Uses OpenZeppelin libraries for string manipulation and mathematical operations.

Functionality

Mint

The users can create new NFTs (pixels) by providing a valid tokenId, a description, and a creator name. The minting process requires payment in Ether, and the payment amount depends on the length of the tokenId.

Upgrades

The users can upgrade an existing token by burning it and minting a new one with a higher-level tokenId. The upgrade process also requires payment in Ether, and the payment

amount is based on the difference in minting prices between the original and upgraded tokens.

Metadata and SVG Generation

- The contract includes a library (`theXandaoMeta`) for generating metadata and SVG content.
- Metadata includes details like creator's address, creator's name, grid size, background color, and composite color.
- SVG content is generated based on `tokenId` and a predefined color map.

Token URI and External URLs

- Functions (`tokenURIByTokenId` and `tokenURIByXN`) retrieve the token URI for a given token.
- External URL (`https://pixels.xandao.com/view?id=`) for viewing pixel art.

Ownership and Withdrawals

- The contract has an owner with special privileges.
- A percentage of Ether from minting and upgrading processes goes to a DAO address and an operations address.

Token and User Information

- Users can query information about tokens, including the creator, availability status, and Ether balance of the contract.
- Mechanism to handle burn status of tokens, indicating whether a token has been upgraded.

External Libraries

The contract uses external libraries, such as OpenZeppelin's ERC-721 and utility libraries for string and math operations.

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	DCGI	Digit Count Gas Inefficiency	Unresolved
	MU	Modifiers Usage	Unresolved
	AISRC	Ascending If Statements Redundant Condition	Unresolved
	CCO	Color Code Optimization	Unresolved
	ECXH	Edge Case XN Handling	Unresolved
	MEM	Misleading Error Messages	Unresolved

DCGI - Digit Count Gas Inefficiency

Criticality	Minor / Informative
Location	XandaoMeta.sol#L51
Status	Unresolved

Description

The current implementation of the `_checkTokenId` function i utilizes string conversion to count the number of digits in a tokenId. Specifically, the function converts `tokenId` to a string and then calculates the length of this string to determine the number of digits. This approach, while functionally correct, is less gas-efficient compared to a direct arithmetic method. String operations in Solidity, particularly conversions and accessing properties like length, are generally more gas-intensive than simple arithmetic operations. The inefficiency arises from the computational overhead associated with string handling and type conversions.

```
//get number of digits
uint8 tokenIdLen = uint8(bytes(tokenId.toString()).length);
```

Recommendation

Consider replacing the current string conversion method with an arithmetic-based method for counting the number of digits in `tokenId` . This change is expected to reduce the gas cost of the function, thereby optimizing the overall contract for more efficient execution and lower transaction fees for users.

MU - Modifiers Usage

Criticality	Minor / Informative
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
uint256 descLen = bytes(description).length;
require(
    descLen < 36 && descLen > 0,
    "Pixels description must be between 1 and 36 characters"
);
...
bool lock = false;
require(!lock);
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

AISRC - Ascending If Statements Redundant Condition

Criticality	Minor / Informative
Status	Unresolved

Description

The contract contains a redundant and unnecessary condition check in the if-else statements within the `getViewBox`, `getGridSize`, and `getBackgroundColor` functions. The conditions for the length of tokenId are ascending, starting from 1 and increasing sequentially. Therefore, it is only required to check if the length is less than the upper bound of each range, as the conditions are mutually exclusive.

```
if (tokenIdLen == 1) {
// ...
} else if (tokenIdLen >= 2 && tokenIdLen <= 4) {
// ...
} else if (tokenIdLen >= 5 && tokenIdLen <= 9) {
// ...
} else if (tokenIdLen >= 10 && tokenIdLen <= 16) {
// ...
} else if (tokenIdLen >= 17 && tokenIdLen <= 25) {
// ...
} else {
// ...
}
```

Recommendation

Simplify the if-else conditions by removing the first condition and retaining only the check for the upper bounds of each range. This enhances code readability and maintains the logical structure of the conditions. The modified conditions would look as follows. This adjustment does not affect the logic of the code but results in cleaner and more concise conditionals.

CCO - Color Code Optimization

Criticality	Minor / Informative
Location	XandaoMeta.sol#L222
Status	Unresolved

Description

The `getColorHSL` function produces HSL values (0, 0, 0), which is the same outcome with the default return value specified in the final else statement of the function. Therefore, the explicit check for "black" is redundant since that case would be adequately handled by the default else condition.

```
function _getColorHSL(uint8 colorIndex) private pure returns
(uint256 h, uint256 s, uint256 l) {
    if (colorIndex == uint8(Colors.CYAN)) {
        return (uint256(180), uint256(100), uint256(50));
    } else if (colorIndex == uint8(Colors.MAGENTA)) {
        return (uint256(300), uint256(100), uint256(50));
    } else if (colorIndex == uint8(Colors.YELLOW)) {
        return (uint256(60), uint256(100), uint256(50));
    } else if (colorIndex == uint8(Colors.BLACK)) {
        return (uint256(0), uint256(0), uint256(0));
    } else if (colorIndex == uint8(Colors.WHITE)) {
        return (uint256(0), uint256(0), uint256(100));
    } else {
        return (uint256(0), uint256(0), uint256(0));
    }
}
```

Recommendation

To enhance the function's efficiency, consider removing the explicit check for "black" color. The default else condition is sufficient for returning the HSL values (0, 0, 0) for that color, as well as any other undefined colors.

ECXH - Edge Case XN Handling

Criticality	Minor / Informative
Location	TheXanDAO.sol#L196
Status	Unresolved

Description

The `upgradePixels` function displays a potential flaw in handling the edge case for `xnVersion` updates. Specifically, the function increments an index to determine the next version of `xnVersion` for a token upgrade. However, this logic does not account for the scenario where the current `xnVersion` is the last element in the `_xnVersions` array. In such a case, incrementing the index would result in an out-of-bounds reference, potentially leading to unexpected behavior or a contract revert due to invalid array access. Moreover, the comment " 'a' " within the code suggests a misunderstanding or an incorrect annotation, as it implies that an index of 0 should correspond to 'a'. In reality, an index of 0 would indicate that the current `xnVersion` is not found in the array, given that the initial index is set to -1 and only updated upon finding a match. This issue highlights a crucial gap in the function's logic, particularly in handling the transition of `xnVersion` when upgrading tokens that have reached the end of the defined version sequence.

```
// get new XN
    int256 idx = -1;
    for (int256 i = 0; i < 36; i++) {
        if (keccak256(abi.encodePacked(XN_VERSIONS[uint256(i)])) ==
keccak256(abi.encodePacked(originTokenObj.xnVersion))) {
            idx = i;
            break;
        }
    }

    uint256 xn = originTokenObj.xn; // keep origin token XN
    uint256 newIdx = uint256(idx + 1);

    string memory xnVersion;  // change XN version
    if (newIdx == 0) { // 'a'
        xnVersion = '';
    } else {
        xnVersion = XN_VERSIONS[newIdx];
    }
```

Recommendation

To address the edge case in the `upgradePixels` function, it is recommended to revise the logic that updates the `xnVersion` for token upgrades. The current implementation should be enhanced to handle the scenario where the `xnVersion` is at the last element of the `XN_VERSIONS` array. Specifically, the function should include a mechanism to reset the `xnVersion` back to the first element in the array when the end is reached. This adjustment would ensure that the function does not attempt to access an out-of-bounds index in the array, thereby preventing unexpected behavior or potential contract reverts. Implementing this change will align the function's behavior with the intended cyclic progression through `xnVersion` values, ensuring reliability and consistency in the token upgrade process.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	TheXanDAO.sol#L154,231
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(!lock)
```

Recommendation

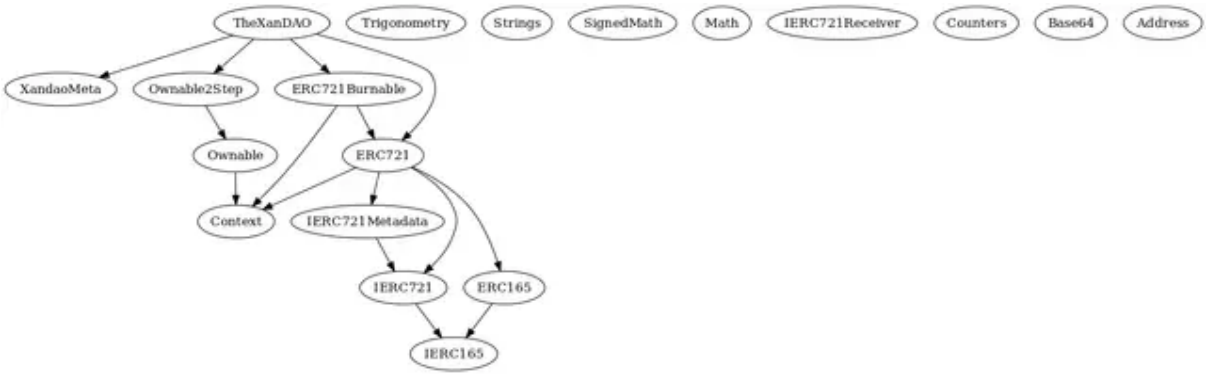
The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

Functions Analysis

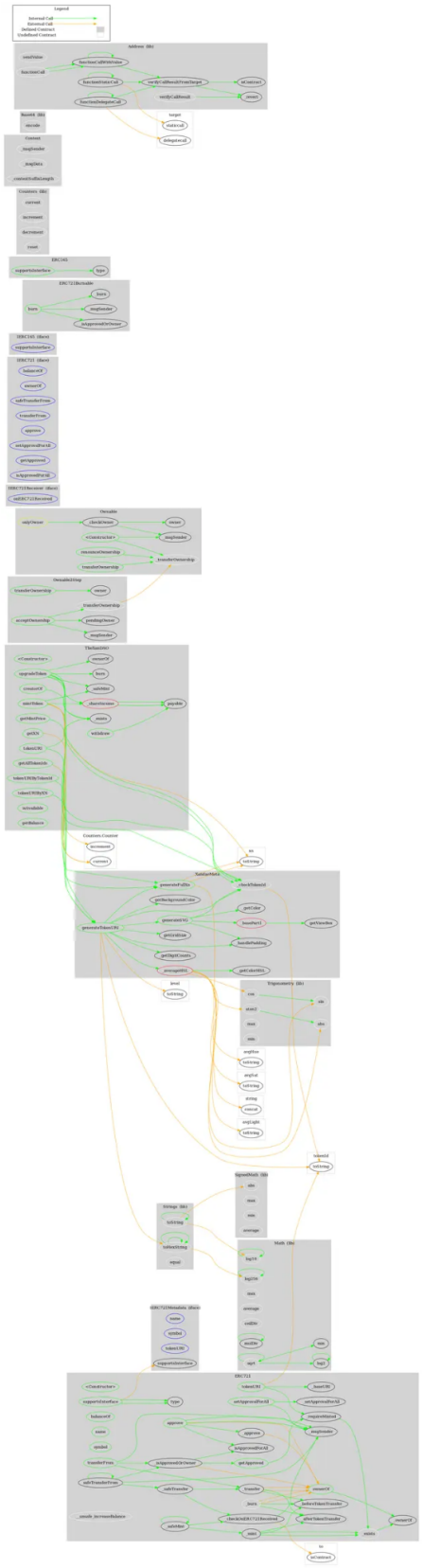
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
XandaoMeta	Implementation			
	_checkTokenId	Internal		
	getGridSize	Public		-
	getViewBox	Public		-
	getBackgroundColor	Public		-
	_basePart1	Private		
	generateSVG	Public		-
	generateFullXn	Public		-
	_handlePadding	Private		
	_getDigitCounts	Private		
	_getColor	Private		
	_getColorHSL	Private		
	_averageHSL	Private		
	generateTokenURI	Public		-
Trigonometry	Library			
	sin	Internal		
	cos	Internal		
	atan2	Internal		

	abs	Internal		
	max	Internal		
	min	Internal		
TheXanDAO	Implementation	ERC721, ERC721Burnable, Ownable2Step, XandaoMeta		
		Public	✓	ERC721
	getXN	Public		-
	getAllTokenIds	Public		-
	getMintPrice	Public		-
	_shareIncome	Private	✓	
	upgradeToken	Public	Payable	-
	mintToken	Public	Payable	-
	tokenURI	Public		-
	tokenURIByTokenId	Public		-
	tokenURIByXN	Public		-
	creatorOf	Public		-
	isAvailable	Public		-
	getBalance	Public		-
	withdraw	Public	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

The XanDAO smart contract serves as the backbone for a collaborative, on-chain, minimalist pixel art NFT project known as Xandao pixels. Enabling users to mint and upgrade pixel art NFTs. Overall, the contract fosters a community-driven exploration of the intersection between art and technology through the lens of pixel art. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io