Audit Report

Myntflo Marketplace

January 2023

Github	https://github.com/oxalexa/myntflo-contracts
Commit	3abe6b1e5bc1534f1aa7c181d8adb574477e3fa2
Audited by	© cyberscope

Table of Contents

Review

Contract Name	MyntfloMarketplace
Repository	https://github.com/oxalexa/myntflo-contracts
Commit	3abe6b1e5bc1534f1aa7c181d8adb574477e3fa2

Audit Updates

Initial Audit	12 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/metatx/ERC2771Context.sol	350e132f5ebc838e000770ceee044e4541a598b05bf998e96285c859eea5d8ef
@openzeppelin/contracts/metatx/MinimalForwarder.sol	95a2f6b10918f410d143f27581a0a1c7603c9dd774c31899bb4cc20cc1619515
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	fd6a1801f1f2f8af0a3ece0b254da06ec24568aec02cfe94827061379aebc6f3
@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol	578834a1bcdac6a22de5e07ae63bbbd4d41615f35950afc6e6c068d92619b334
@openzeppelin/contracts/token/ERC1155/utils/ERC1155Holder.sol	a7ad38fa0a06fe6e24f81fee4f1fc3870767db96d1ba37df7be1199f7a3ace7f
@openzeppelin/contracts/token/ERC1155/utils/ERC1155Receiver.sol	cf407886a0ce7e2af7efe7867e2d2864903426f63eeaa68eecf33d57f7d910c2
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC721/IERC721.sol	fde830ac73ef320f7e3ce977b8cf567173f1e479ba86d584498f8362a67a5dc0
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	77f0f7340c2da6bb9edbc90ab6e7d3eb8e2ae18194791b827a3e8c0b11a09b43
@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol	2cfe4ed66b63283ca12b0360b1c8c1a3c298a510e2e29c60d9ccedb634b738a6
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol	fc0e6c5d7184bd03b8deae6ca9a48a1eaaecf9f5e4703611aabfb63401e6d43f

@openzeppelin/contracts/utils/cryptography/ECDSA.sol	4e45d53327d561848fbcf381262ec5c0ac91b2f1f06432210bf76db55279d945
@openzeppelin/contracts/utils/introspection/ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca0dfe5ec14916369a2935d888ff257a
@openzeppelin/contracts/utils/Strings.sol	34127ad0054df5963b0fd694c1b313d17e9114a2f426b85526d6d976210298ab
contracts/testingDeploy/MyntfloMarketplace.sol	47b32ed062bfa99280830de0c88d360ee7fa0bcf3146e96b4d1b7dbe2918caa1

Introduction

MyntfloMarketplace implements marketplace/exchange mechanism. The contract owner deposits ERC1155 and ERC721 tokens to the contract and sets a specific price per token. Then the users have the ability to buy each of those tokens by providing the "paymentToken" as currency. The "paymentToken" can be configured by the contract owner.

Roles

Owner

- setPaymentToken()
- addERC1155()
- removeERC1155()
- addERC721()
- removeERC721()

Users

buyToken()

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PTC	Potential TokenId Conflict	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

PTC - Potential TokenId Conflict

Criticality	Minor / Informative
Status	Unresolved

Description

The contract combines two different token protocols in one collection. The protocols are ERC721 and ERC1155. Both protocols are indexing the corresponding tokens with a tokenId. Since these two protocols consist of different instances, then the same tokenId value could index a token from both collections. The contract does not deterministically guarantee the uniqueness of the tokenId. The token relies on the fact that the contract owner will provide the proper `contractAddress` that will distinguish the potential conflict.

```
function findListingByTokenId(uint256 tokenId, address contractAddress) public
view returns (bool, uint256) {
    for (uint256 i = 0; i < listings.length; i++) {
        if (listings[i].tokenId == tokenId && listings[i].contractAddress ==
contractAddress) {
            return (true, i);
        }
    }
    return (false, 0);
}
```

Recommendation

The team is advised to add an additional boolean indicator (`isERC1155`) in the `findListingByTokenId()` method. That way the contract will guarantee that even if the owner misuses the method parameters, there will not be a conflict between the two protocols.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/testingDeploy/MyntfloMarketplace.sol#L68
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
IERC20 _paymentToken
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/testingDeploy/MyntfloMarketplace.sol#L2
Status	Unresolved

Description

The ^ symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.13;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ERC2771Context	Implementation	Context		
		Public	✓	-
	isTrustedForwarder	Public		-
	_msgSender	Internal		
	_msgData	Internal		
MinimalForwarder	Implementation	EIP712		
		Public	✓	EIP712
	getNonce	Public		-
	verify	Public		-
	execute	Public	Payable	-
IERC1155	Interface	IERC165		
	balanceOf	External		-
	balanceOfBatch	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-
	safeBatchTransferFrom	External	✓	-
IERC1155Receiver	Interface	IERC165		
	onERC1155Received	External	✓	-
	onERC1155BatchReceived	External	✓	-

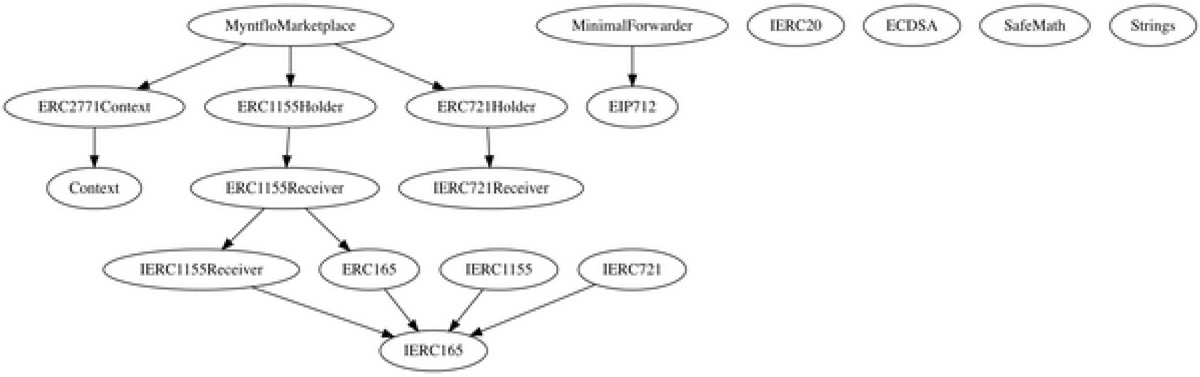
ERC1155Holder	Implementation	ERC1155Receiver		
	onERC1155Received	Public	✓	-
	onERC1155BatchReceived	Public	✓	-
ERC1155Receiver	Implementation	ERC165, IERC1155Receiver		
	supportsInterface	Public		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC721	Interface	IERC165		
	balanceOf	External		-
	ownerOf	External		-
	safeTransferFrom	External	✓	-
	safeTransferFrom	External	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	setApprovalForAll	External	✓	-
	getApproved	External		-
	isApprovedForAll	External		-
IERC721Receiver	Interface			

	onERC721Received	External	✓	-
ERC721Holder	Implementation	IERC721Receiver		
	onERC721Received	Public	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
EIP712	Implementation			
		Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		
ECDSA	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toTypedDataHash	Internal		
ERC165	Implementation	IERC165		
	supportsInterface	Public		-

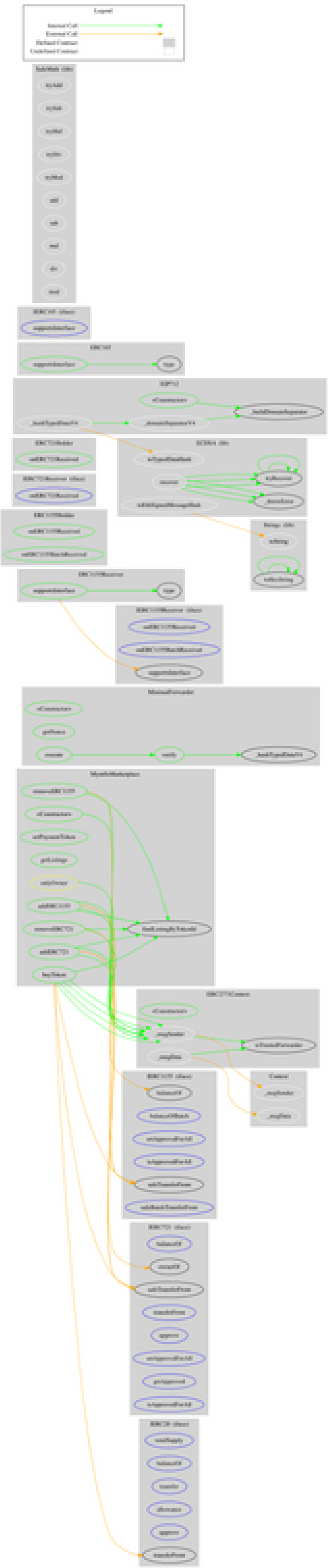
IERC165	Interface			
	supportsInterface	External		-
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
MyntfloMarketplace	Implementation	ERC2771Context, ERC1155Holder, ERC721Holder		
		Public	✓	ERC2771Context

	setPaymentToken	Public	✓	onlyOwner
	getListings	Public		-
	findListingByTokenId	Public		-
	addERC1155	Public	✓	onlyOwner
	removeERC1155	Public	✓	onlyOwner
	addERC721	Public	✓	onlyOwner
	removeERC721	Public	✓	onlyOwner
	buyToken	Public	✓	-

Inheritance Graph



Flow Graph



Summary

MyntfloMarketplace contract implements a marketplace mechanism. 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