Cyberscope

Audit Report

Altair Factory

December 2022

SHA256 3f48a9538cefcfc660714bcd27b9964aaf9736b0ec0dbe4d7c1163db49b7351e

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	AltairFactory
Testing Deploy	https://testnet.bscscan.com/address/0x0b87895c0626db2d33eeee0dd91653de3f2863fd

Audit Updates

Initial Audit	16 Nov 2022 https://github.com/cyberscope-io/audits/blob/main/nali/v1/altairFactory.pdf
Corrected Phase 2	9 Dec 2022 https://github.com/cyberscope-io/audits/blob/main/nali/v1/altairFactory.pdf
Corrected Phase 3	22 Dec 2022

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	da66c17044345dc892d85bd7ddc9745d25df0b3dacfba8f84eb87c60d6e40fe3
@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol	424a35f695e5cb428acb770642b1370d38443fe74edf18da786ee9486c47ea23
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	74def996fd6faf32f13ab9cacfc71d57400177de340fe5d5d7c6e805dfbab3bd
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	c05b019a0b3bee8f3fac2da7c929f7d665b97d6d046aa35126615fff11205119
@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol	8b4e2fd7e04b94b6717095b54aeb001f75e49b0e974b04a41ea2e7fb742877ea
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155BurnableUpgradeable.sol	4cef49efc3c1d62289ff13b719092fd0191fd979448147f2b7588de4cea2cfae
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155SupplyUpgradeable.sol	98d3570efe134810096cab2ce5d39c51323d9216e8528198fac44dd15cd3c841
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155URIStorageUpgradeable.sol	7da34d679483559e38cbae26e66e21d39b4e5f191b1942669b45ac29cd965f09
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/IERC1155MetadataURIUpgradeable.sol	61e3317af2516530f091665d198fc3e27fb514038600fb07b7813913f439775f
@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155ReceiverUpgradeable.sol	7108589dbf9528c2ffe4f17fe489e8183be55c15b51af3b88c70252c13bac539
@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155Upgradeable.sol	5321f224c08b59651968dde0db5abeec4ea0bdf371c7b0c25707e56b455882fe
@openzeppelin/contracts-upgradeable/token/ERC1155/utils/ERC1155HolderUpgradeable.sol	fe3758383dfee87be40050cc903408f0a4c7f20eb430931dcbb8fa337cbdfa4f
@openzeppelin/contracts-upgradeable/token/ERC1155/utils/ERC1155ReceiverUpgradeable.sol	59bd51ef07cf35a5de2e0090231ed4398d529173a2749c04fb04874613b3da50
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol	b97515a88e75c313eacf0a27c9439ef371d86d4c2730d3b13076640942f813df

@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	4e09a7479aa3e7c313f8fc141c4c8fc04e0abfeb8754615ef7d78ec94c298b07
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	45b47dd617d02875a7e6c896d1842ff9d8362ab15b8180645f3f4b180d4f028f
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	1d7d481b79fd54d957c9a0696f6227f7799fec01d8ba41f5c130a7cc6b4eddc9
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	fd84e5284eccc479268f0ef36b830019d4f7999ceb7959430d8d8d9e602dd4ef
@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol	a39bc026ad6214e9ecd526bd4a1ddf9862d80bd4a9d0d031d9bafa4c3c147c0b
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	158a0316fa289fad12c2ca764449e43e6724fb79c58fc438508d116f9af46b39
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	68f5690fc266a6b48386c28cbfd72ec67c24b05a51ce26d24103577c15f61401
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	73f4fcf0069fbc82f59bcc935b066788a485451381903487a251fc255b78f314
contracts/AltairFactory.sol	3f48a9538cefcfc660714bcd27b9964aaf9736b0ec0dbe4d7c1163db49b7351e
contracts/interfaces/IAltairFund.sol	30f415f676820b6f9f005e1022eb7c23ac5c7e211640919f2c2bf9a71a64dd98
contracts/interfaces/IAltairMaster.sol	d441fc5c3c90f5557ca8715269d3fc23f9aaae007ff8954dd61a5f9d8599cd25
contracts/interfaces/IAltairShop.sol	e4bbaca996c77e95097b2641444dbb32a777067b60e23b763c3fe7f7b48b8c0b
contracts/interfaces/IAltairSwap.sol	0b2a9ce80407df392c551deb6179f0031a9d69a0a227622c26bd68eaa925b7f1
contracts/interfaces/IAltairSwapInfo.sol	dc46d60af278bfd40dce138d166d2dde275678009060baa3c9851bd68cfe60d7
contracts/interfaces/IContractInitializer.sol	7606aa126257e99dca9aec4b61997d5886dd0b4960a04c0b710cf6dc5c5d373a
contracts/interfaces/INaliToken.sol	b516894970890c70ee7c748285b79adca4c45c34e6f6822a1de08b1843ba14f6

contracts/interfaces/IProxyCall.sol	bd6f89340af3460c079dd3853d2e1a87c1e02ae124b3b732b83ef3b32deac6bd
contracts/interfaces/IRoles.sol	d13aac5175ddfd3af1ee4dc652d46b5f317214c0455a10ee580c0b2d414876e0
contracts/interfaces/ITreasury.sol	23176edee2b0b416db08b8bd686821cd51afdeb015f8e97a85a83141b8134315
contracts/interfaces/IWETH.sol	4b36d18ea8a606a912484eef60d6e2c5e4c3baa8ac64c3541775394fa1fd633e
contracts/libs/AltairLib.sol	95ae3593a0abb118bb7cf4670fb28b119873b8a7861ceb6cd39d9b099bb3bb30
contracts/libs/SwapData.sol	c4b79ab1f23186ae4c19b628a5cb4019c5308c109bb707f553360ab3ff06ca46
contracts/NaliAltair.sol	13d88ae3af117596663e5cc000fff9f31ee862418f9c1091717bf0786a2f08b4

Introduction

The Altair Factory ecosystems contracts are implemented as an upgradable proxy. It consists of AltairFund, AltairMaster, and AltairFactory contracts.

AltairFactory

The Altair Factory is responsible for creating funds, and managing the platform's parameters. Additionally, the contract is responsible for minting, burning Nali NFTs, and subscribing to an AltainFund.

AltairFund

The Altair Factory implements a pooling mechanism for the created funds. Users can subscribe to a fund, redeem, and emergencyRedeem their investment.

AltairMaster

The AltairMaster is responsible for providing information from the world outside to the blockchain. To be more specific the AltairMaster contract keeps info about tokens prices from the liquidity pool. Additionally, the contract provides price data from three different Oracles. Chainlinks, Band, and 1-inch oracle.

Roles

The contract roles are provided by an outside source. The Roles contract is out of the scope of this audit.

AltairFactory

The AltairFactory contract has an admin role.

Admin

The admin has the authority to

- Change CustomToken address.
- Change Native address.
- Change NaliToken address.
- Change subscription fee.
- Change Redeem fee.
- Change fee variable.
- Set UseSwapInfo.
- Change Manager fee.
- Set Treasury address.
- Set Master address.
- Set Shop address.
- Set AltairSwap address.
- Set SwapContract address.
- Change Monthly cost.
- Change the Free Trial period.
- Update contract implementation.
- Update proxy call contract
- Update Roles contract.
- Change the Default Operator address.

AltairMaster

The AltairMaster contract has an admin role.

Admin

The admin has the authority to

- Change the router address.
- Update tokens name.
- Add PancakePriceToken.
- Update Supporting Fee On TransferTokens.

AltairFund

The AltairFund contract has a manager role and a factory role.

Manager

The manager role consists of the managerOwner and the authorized users. The manager role has the authority to

- Set copyTrading.
- Set DCA.
- Set Manager monthly fee.
- Redeem the manager's monthly fee.
- Pause or Unpause funds subscriptions.
- Create TargetNames.
- Update rebalance period.
- Update manager property.
- Update Non-Balance Manager Names
- Rebalance.
- Can swap non-index tokens to BNB.
- Set Authorized address.

Factory

The factory roles have the authority to

- Update the platform address.
- Update max manager monthly fee.

Contract Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	RFS	Redundant For-Loop Statement	Unresolved
	USV	Unused State Variables	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L14	Uninitialized Variables in Local Scope	Unresolved
	L15	Local Scope Variable Shadowing	Unresolved
	L18	Multiple Pragma Directives	Unresolved
	L22	Potential Locked Ether	Unresolved

RFS - Redundant For-Loop Statement

Criticality	Minor / Informative
Location	contracts/AltairFactory.sol#L482,506,526
Status	Unresolved

Description

The contract is utilizing a redundant for-loop statements. The contract is utilizing the for-loop statement after an array construction. When an array has been constructed the length of the array is always zero. Hence the for-loop statement is redundant.

```
function updateAllMaxManagerMonthlyFee(
    address[] memory fundsArray,
    uint256 feeAmt
) external onlyAdmin returns (bool) {
    fundsArray = funds.values();

    uint256 fundsLength = fundsArray.length;

    if (fundsArray.length == 0) {
        fundsArray = new address[](fundsLength);

        for (uint256 i; i < fundsLength; i++) {
            fundsArray[i] = funds.at(i);
            IAltairFund(fundsArray[i]).updateManagerMaxMonthlyFee(feeAmt);
        }
    } else {
        for (uint256 i; i < fundsArray.length; i++) {
            IAltairFund(fundsArray[i]).updateManagerMaxMonthlyFee(feeAmt);
        }
    }

    return true;
}

function updateAllPlatformAddresses(address[] memory fundsArray)
    external
    onlyAdmin
    returns (bool)
{
    if (fundsArray.length == 0) {
        fundsArray = funds.values();
    } else {
        fundsArray = fundsArray;
    }

    uint256 fundsLength = fundsArray.length;

    for (uint256 i; i < fundsLength; i++) {
        IAltairFund(fundsArray[i]).updatePlatformAddresses();
    }

    return true;
}

function updateAllCopyTrading(address[] memory fundsArray)
    external
    returns (bool)
{
    if (fundsArray.length == 0) {
        fundsArray = funds.values();
```

```
    } else {
        fundsArray = fundsArray;
    }

    uint256 fundsLength = fundsArray.length;

    for (uint256 i; i < fundsLength; i++) {
        if (IAltairFund(fundsArray[i]).copytrading()) {
            IAltairFund(fundsArray[i]).updateCopyTrading();
        }
    }

    return true;
}
```

Recommendation

The contract should use for-loop statements only when then the array length `fundsArray.length` is greater than 0.

USV - Unused State Variables

Criticality	Minor / Informative
Status	Unresolved

Description

The contract is processing variables that are not utilized in the contract's implementation.

```
IAltairFactory factory;
```

Recommendation

The contract should remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/libs/AltairLib.sol#L43 contracts/AltairFactory.sol#L75,91
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 your 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 your code.
7. Keep lines short (around 120 characters) to improve readability.

```
struct transferData {
        address[] targetNamesAddress;
        uint256 totalTrfAmt;
        uint256 totalUnderlying;
        uint256 qtyToTrfAToken;
    }

...
```

Recommendation

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

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

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/AltairFactory.sol#L474,493,498,519,538
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in your contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 i
```

Recommendation

By initializing local variables before using them, you can help ensure that your contract functions behave as expected and avoid potential issues.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	contracts/interfaces/IAltairFund.sol#L47
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
address managerOwner
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/NaliAltair.sol#L2 contracts/libs/SwapData.sol#L1 contracts/libs/AltairLib.sol#L1 contracts/interfaces/IWETH.sol#L1 contracts/interfaces/ITreasury.sol#L2 contracts/interfaces/IRoles.sol#L3 contracts/interfaces/IProxyCall.sol#L3 contracts/interfaces/INaliToken.sol#L3 contracts/interfaces/IContractInitializer.sol#L2 contracts/interfaces/IAltairSwapInfo.sol#L2 contracts/interfaces/IAltairSwap.sol#L3,4 contracts/interfaces/IAltairShop.sol#L1 contracts/interfaces/IAltairMaster.sol#L3,4 contracts/interfaces/IAltairFund.sol#L2 contracts/AltairFactory.sol#L2,3
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.0;
pragma solidity ^0.8.2;
pragma solidity ^0.8.1;
pragma solidity 0.8.14;
pragma experimental ABIEncoderV2;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, you can avoid conflicts and ensure that the contract can be compiled correctly.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	contracts/AltairFactory.sol#L628
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
receive() external payable {}
```

Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal	✓	onlyInitializing
	__Ownable_init_unchained	Internal	✓	onlyInitializing
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
	_disableInitializers	Internal	✓	
	_getInitializedVersion	Internal		
	_isInitializing	Internal		
PausableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Pausable_init	Internal	✓	onlyInitializing
	__Pausable_init_unchained	Internal	✓	onlyInitializing
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused

ERC1155Upgradeable	Implementation	Initializable, ContextUpgradeable, ERC165Upgradeable, IERC1155Upgradeable, IERC1155MetadataURIUpgradeable		
	__ERC1155_init	Internal	✓	onlyInitializing
	__ERC1155_init_unchained	Internal	✓	onlyInitializing
	supportsInterface	Public		-
	uri	Public		-
	balanceOf	Public		-
	balanceOfBatch	Public		-
	setApprovalForAll	Public	✓	-
	isApprovedForAll	Public		-
	safeTransferFrom	Public	✓	-
	safeBatchTransferFrom	Public	✓	-
	_safeTransferFrom	Internal	✓	
	_safeBatchTransferFrom	Internal	✓	
	_setURI	Internal	✓	
	_mint	Internal	✓	
	_mintBatch	Internal	✓	
	_burn	Internal	✓	
	_burnBatch	Internal	✓	
	_setApprovalForAll	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
	_doSafeTransferAcceptanceCheck	Private	✓	
	_doSafeBatchTransferAcceptanceCheck	Private	✓	

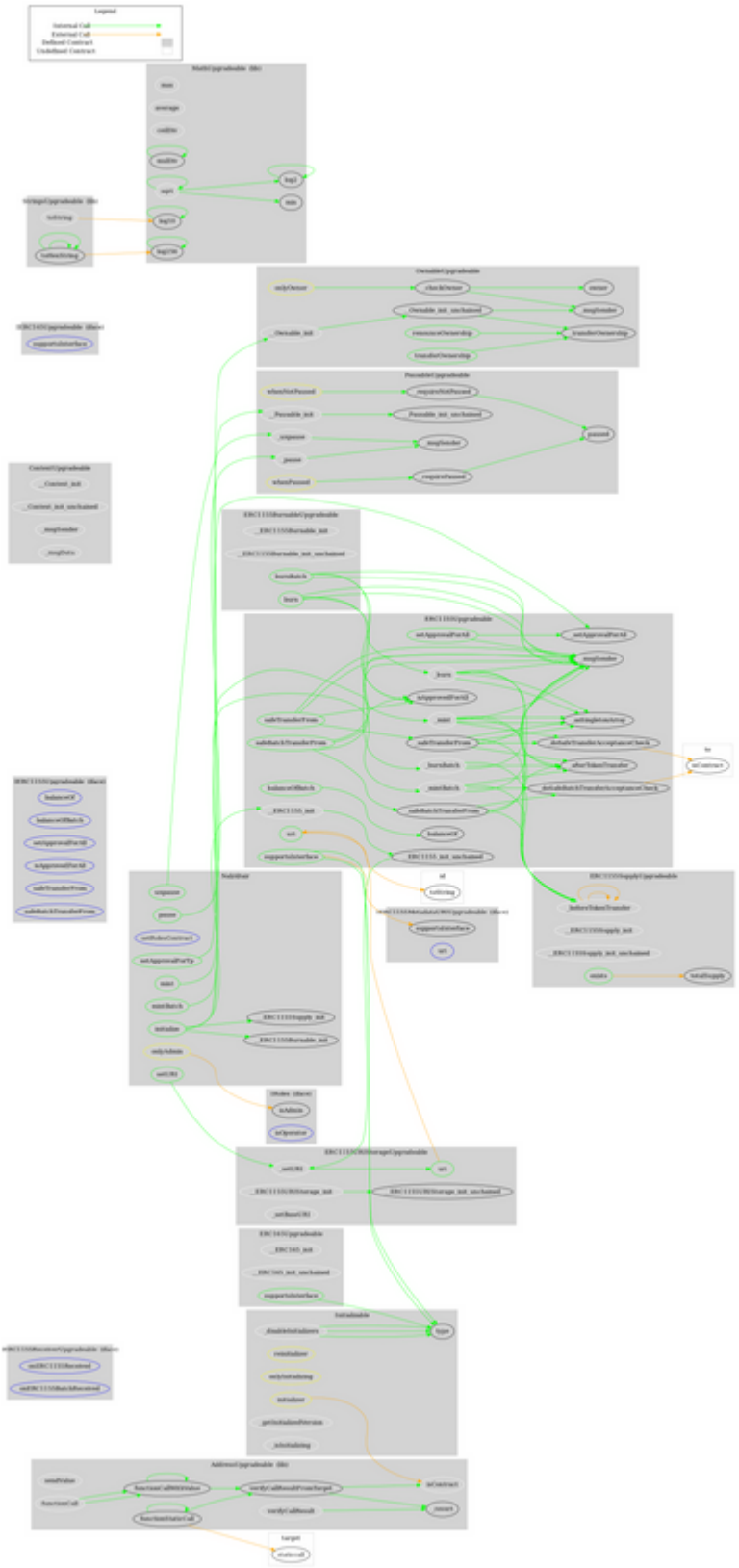
	_asSingletonArray	Private		
ERC1155BurnableUpgradeable	Implementation	Initializable, ERC1155Upgradeable		
	__ERC1155Burnable_init	Internal	✓	onlyInitializing
	__ERC1155Burnable_init_unchained	Internal	✓	onlyInitializing
	burn	Public	✓	-
	burnBatch	Public	✓	-
ERC1155SupplyUpgradeable	Implementation	Initializable, ERC1155Upgradeable		
	__ERC1155Supply_init	Internal	✓	onlyInitializing
	__ERC1155Supply_init_unchained	Internal	✓	onlyInitializing
	totalSupply	Public		-
	exists	Public		-
	_beforeTokenTransfer	Internal	✓	
ERC1155URIStorageUpgradeable	Implementation	Initializable, ERC1155Upgradeable		
	__ERC1155URIStorage_init	Internal	✓	onlyInitializing
	__ERC1155URIStorage_init_unchained	Internal	✓	onlyInitializing
	uri	Public		-
	_setURI	Internal	✓	
	_setBaseURI	Internal	✓	
IERC1155MetadataURIUpgradeable	Interface	IERC1155Upgradeable		
	uri	External		-

IERC1155ReceiverUpgradeable	Interface	IERC165Upgradeable		
	onERC1155Received	External	✓	-
	onERC1155BatchReceived	External	✓	-
IERC1155Upgradeable	Interface	IERC165Upgradeable		
	balanceOf	External		-
	balanceOfBatch	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-
	safeBatchTransferFrom	External	✓	-
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResultFromTarget	Internal		
	verifyCallResult	Internal		
	_revert	Private		
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing

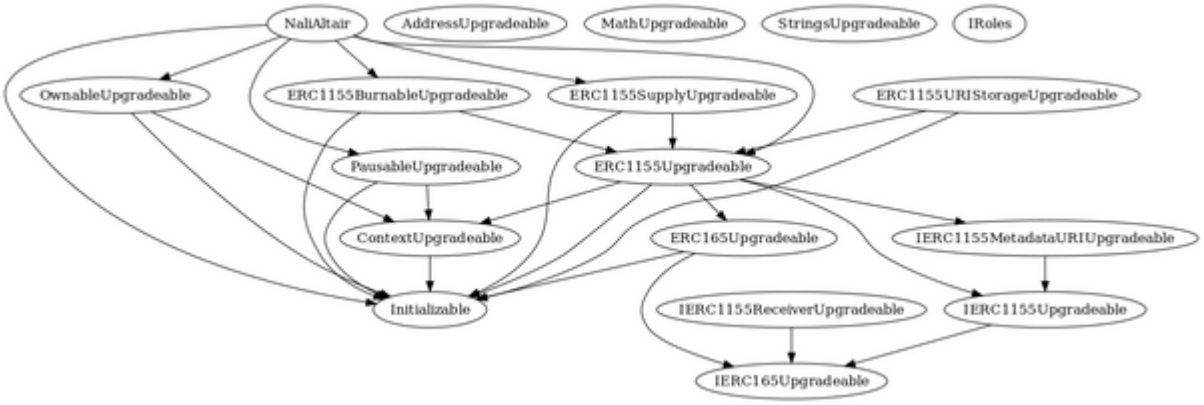
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		
ERC165Upgradeable	Implementation	Initializable, IERC165Upgradeable		
	__ERC165_init	Internal	✓	onlyInitializing
	__ERC165_init_unchained	Internal	✓	onlyInitializing
	supportsInterface	Public		-
IERC165Upgradeable	Interface			
	supportsInterface	External		-
MathUpgradeable	Library			
	max	Internal		
	min	Internal		
	average	Internal		
	ceilDiv	Internal		
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		
	sqrt	Internal		
	log2	Internal		
	log2	Internal		
	log10	Internal		
	log10	Internal		
	log256	Internal		
	log256	Internal		

StringsUpgradeable	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
IRoles	Interface			
	isAdmin	External		-
	isOperator	External		-
NaliAltair	Implementation	Initializable, ERC1155Upgradeable, OwnableUpgradeable, PausableUpgradeable, ERC1155BurnableUpgradeable, ERC1155SupplyUpgradeable		
	initialize	Public	✓	initializer
	setRolesContract	External	✓	onlyAdmin
	uri	Public		-
	setURI	Public	✓	onlyAdmin
	pause	Public	✓	onlyAdmin
	unpause	Public	✓	onlyAdmin
	mint	Public	✓	onlyAdmin
	mintBatch	Public	✓	onlyAdmin
	_beforeTokenTransfer	Internal	✓	whenNotPaused
	setApprovalForTp	Public	✓	onlyAdmin

Contract Flow



Inheritance Graph



Summary

The Altair Factory ecosystem operates as a funds investment mechanism. This audit focused on investigating possible 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