Cyberscope

Audit Report

Altair Factory

December 2022

SHA256 09dfd6013ceafce40f667e5ae980d72a510fa9df6f44425bde3f18d8db5ade8c

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	AltairFactory
Compiler Version	v0.8.14+commit.80d49f37
Testing Deploy	https://testnet.bscscan.com/address/0x752F70f7f261C2987263593FFCF8b464755f5db9#code
Domain	https://nali.finance

Audit Updates

Initial Audit	16th November 2022 https://github.com/cyberscope-io/audits/blob/main/nali/v1/altairFactory.pdf
Corrected	9th December 2022

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	da66c17044345dc892d85bd7ddc9745d25df0b3dacfba8f84eb87c60d6e40fe3
@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol	11d95e1dddcd4351da5eb6eb4b1d6fa0f30afc45e4bad6d34028185a28a1cf65
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	cd823c76cbf5f5b6ef1bda565d58be66c843c37707cd93eb8fb5425deebd6756
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	c05b019a0b3bee8f3fac2da7c929f7d665b97d6d046aa35126615fff11205119
@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol	7b5314b4e3ddc497ba6bb51c783cbc38762526cd7e801fbc28ba9e00317c2a76
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155BurnableUpgradeable.sol	80c8ff8a46a2b197bee6518186627db7dda386b4a840aa162d32257bc97c9fff

@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	b7410d275fc7d26e36b0851541d6ff290593ba72d64b5c906978124b123915c1
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	35fb271561f3dc72e91b3a42c6e40c2bb2e788cd8ca58014ac43f6198b8d32ca
@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/StringsUpgradeable.sol	e7b950eee23563e23989a3b51a1456614a1838084eef1fad04eb2be0bc280f48
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	014846dc6c387e8fd6d31df636c28898eed601dc668725edbb1a0606c58d4b7e
contracts/AltairFactory.sol	7b47219d9ce4937d6c96752f2d5e5eb49b00c61684a80d591566002127c058f9
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	91b888f0eb5deb765df874397e0c88740e14f9037a37ecdb4bf774a77a9fa3f1

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
●	ZD	Zero Division	Unresolved
●	RSI	Redundant Statements Issue	Unresolved
●	USV	Unused State Variables	Unresolved
●	RAI	Redundant Arguments Issue	Unresolved
●	MTP	Misuse of Transaction Property	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

ZD - Zero Division

Criticality	minor / informative
Location	contract.sol/AltairFund.#L493,764,1076,1082
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

```
uint256 redeemratio = tp.amount.mul(1e18).div(totalSupply());
uint256 redeemratio = tradeParams.amount.mul(1e18).div(totalSupply());
fundWeight = destValue.mul(10000).div(totalfundvalue);
uint256 price = _getLatestPrice(destAddress);
destRebQty =
destActiveWeight.mul(totalfundvalue).mul(1e18).div(price).div(10000);
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow them to execute the corresponding statements.

RSI - Redundant Statements Issue

Criticality	minor / informative
Location	contract.sol/AltairFactory.sol#L513,538
Status	Unresolved

Description

The contract is processing 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 updateAllPlatformAddresses(address[] memory fundsArray)
        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]).updatePlatformAddresses();
            }
        } else {
            for (uint256 i; i < fundsArray.length; i++) {
                IAltairFund(fundsArray[i]).updatePlatformAddresses();
            }
        }

        return true;
    }

    function updateAllCopyTrading(address[] memory fundsArray)
        external
        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);

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

        return true;
    }
```

Recommendation

The contract should remove redundant for-loop statements.

USV - Unused State Variables

Criticality	minor / informative
Location	contract.sol/AltairMaster.sol#L29
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.

RAI - Redundant Arguments Issue

Criticality	minor / informative
Location	contract.sol/AltairFactory.sol#L158,173,195
Status	Unresolved

Description

The contract is processing an argument to illustrate the functions msg.sender. Since the contract could utilize the msg.sender the argument is redundant.

```
    function mint(
        address fund,
        address account,
        uint256 amount,
        bytes memory data
    ) external returns (bool) {
        require(msg.sender == fund, "Sender not fund");

        uint256 id = getFundId(msg.sender);

        naliAltair.mint(account, id, amount, data);

        return true;
    }

    function burn(
        address fund,
        address account,
        uint256 amount
    ) external returns (bool) {
        require(msg.sender == fund, "Sender not fund");

        require(funds.contains(msg.sender));

        bool isApproved = naliAltair.isApprovedForAll(account, address(this));

        if (!isApproved) {
            naliAltair.setApprovalForTp(account, address(this), true);
        }

        uint256 _id = getFundId(msg.sender);
```

```
        naliAltair.burn(account, _id, amount);

        return true;
    }

    function subscribe(address fund, bool useNali) external returns (bool) {
        require(funds.contains(fund));

        if (subscriptions[fund] < block.timestamp) {
            subscriptions[fund] = block.timestamp + 30 days;
        } else {
            subscriptions[fund] += 30 days;
        }
        if (useNali) {
            bool success = naliToken.transferFrom(
                msg.sender,
                address(treasury),
                monthlyCost
            );
            require(success, "Nali transfer failed");
        } else {
            bool success = customToken.transferFrom(
                msg.sender,
                address(treasury),
                monthlyCost
            );
            require(success, "Custom token transfer failed");
        }

        emit Subscribed(fund);

        return true;
    }
```

Recommendation

The contract should only use the transaction property msg.sender and not incorporating extra function arguments.

MTP - Misuse of Transaction Property

Criticality	minor / informative
Location	contract.sol/AltairFund.sol#824
Status	Unresolved

Description

The contract is processing an argument to illustrate the functions msg.sender.

```
    function emergencyRedeem(uint256 redeemUnit, address investorAddress)
        external
        payable
    {
        uint256 totalValueB4 = _getFundValues();

        uint256 redeemratio = redeemUnit.mul(1e18).div(totalSupply());
        require(redeemratio > 0, "MustBeZero");
        factory.burn(address(this), msg.sender, redeemUnit);
        uint256 totalBaseBal = address(this).balance;
        uint256 totalOutput = redeemratio.mul(totalBaseBal).div(1e18);
        TransferHelper.safeTransferBNB(investorAddress, totalOutput);

        for (uint256 i; i < targetNamesAddress.length; i++) {
            AltairLib.transferData memory _transferData = _getTransferAmt(
                targetNamesAddress[i],
                redeemratio
            );
            if (_transferData.totalTrfAmt > 0) {
                TransferHelper.safeTransfer(
                    targetNamesAddress[i],
                    investorAddress,
                    _transferData.totalTrfAmt
                );
            }
        }

        uint256 totalValue = _getFundValues();

        emit EmergencyRedeemComplete(totalValueB4, totalValue, true);
    }
```

Recommendation

The contract should only use the transaction property msg.sender.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contracts/AltairFund.sol#L67
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
EMPTY_DATA
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/AltairFactory.sol#L91,54,75,110 contracts/AltairMaster.sol#L239,238,33 contracts/AltairFund.sol#L561,625,37,71,34,553,626,38,67,562
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
redeemFeeUpdated
NATIVE
fundCreated
NATIVECtx

_useSupportingFeeOnTransferTokens
_tokenaddress
TokenNames

_toAddresses
_deadline
WETH
TargetWeight
BaseTokenName
_pause
_priceImpactTolerance
BaseToken
EMPTY_DATA
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contracts/AltairFactory.sol#L538 contracts/AltairFund.sol#L202,138,341,690,560,997,312,402
Status	Unresolved

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
IAltairFund(fundsArray[i_scope_0]).copytrading() == true
IAltairFund(fundsArray[i]).copytrading() == true

nonBMNamesMapping[nonBMNamesAddress[i]] == true
require(bool)(msg.sender == managerOwner || authorized[msg.sender] == true)
managerUnit > 0 && copytrading == false
dca == false
require(bool,string)(copytrading == false,CopyActive)
managerUnit > 0 && copytrading == true
nonBMNamesMapping[nonBMNamesAddress[i_scope_0]] == true
require(bool,string)(pause == false,Pause)
isCopy == false
...
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor / informative
Location	contracts/AltairMaster.sol#L192 contracts/AltairFund.sol#L756,470,824,1184
Status	Unresolved

Description

Performing divisions before multiplications may cause lose of prediction.

```
lpTokenPrice = tokenReserveCumulative.div(totalSupply).mul(px1).mul(2)

redeemratio = tradeParams.amount.mul(1e18).div(totalSupply())
redeemratio = tp.amount.mul(1e18).div(totalSupply())
redeemratio = redeemUnit.mul(1e18).div(totalSupply())
rebaseActiveWgt =
underweightNames[i].activeWeight.mul(10000).div(totalunderActiveweight)
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contracts/AltairFactory.sol#L525,481,505,530,549,500,557 contracts/AltairMaster.sol#L102,92,24 contracts/AltairFund.sol#L204,908,781,230,1041,607,1265,613,1253,1192,505,726,1000,1166,910,1259,1279,837,1004,1131,486
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
i
I_scope_0

i

i
_transferData
i_scope_1
i_scope_0
_underWeightData
x
tp
...
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contracts/interfaces/IAltairFund.sol#L47 contracts/AltairFund.sol#L441,928
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
managerOwner

dca
totalSupply
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
INaliNftTicket	Interface			
	canAccess	External		-
AltairFactory	Implementation	Initializable, ERC1155HolderUpgradeable		
	initialize	Public	✓	initializer
	getFundId	Public		-
	totalSupply	External		-
	mint	External	✓	-
	burn	External	✓	-
	subscribe	External	✓	-
	changeCustomToken	External	✓	onlyAdmin
	changeCustomNative	External	✓	onlyAdmin
	changeNaliTokens	External	✓	onlyAdmin
	changeSubscribeFee	External	✓	onlyAdmin
	changeRedeemFee	External	✓	onlyAdmin
	changeFee	External	✓	onlyAdmin
	setUseSwapInfo	External	✓	onlyAdmin
	changeManagerFee	External	✓	onlyAdmin
	setTreasury	External	✓	onlyAdmin
	setMaster	External	✓	onlyAdmin
	setShop	External	✓	onlyAdmin
	setAltairSwap	External	✓	onlyAdmin
	setSwapContract	External	✓	onlyAdmin
	getFee	External		-
	getManagerFee	External		-
	changeMonthlyCost	External	✓	onlyAdmin

	changeFreeTrial	External	✓	onlyAdmin
	adminUpdateImplementation	External	✓	onlyAdmin
	adminUpdateProxyCallContract	External	✓	onlyAdmin
	adminUpdateRolesContract	External	✓	onlyAdmin
	create	External	✓	-
	getVersion	Public		-
	fundAt	External		-
	containsFund	External		-
	numberOfFunds	External		-
	getFunds	External		-
	updateAllMaxManagerMonthlyFee	External	✓	onlyAdmin
	updateAllPlatformAddresses	External	✓	onlyAdmin
	updateAllCopyTrading	External	✓	-
	_updateRolesContract	Private	✓	
	_updateProxyCallContract	Private	✓	
	_updateImplementation	Private	✓	
	_getSalt	Private		
	onERC1155Received	Public	✓	-
	onERC1155BatchReceived	Public	✓	-
	changeDefaultOperator	External	✓	onlyAdmin
	<Receive Ether>	External	Payable	-
AltairFund	Implementation	Initializable, OwnableUpgradeable, ReentrancyGuardUpgradeable		
	onlyManager	Internal		
	onlyFactory	Internal		
	initialize	Public	✓	initializer
	<Receive Ether>	External	Payable	-
	getTransferAmt	Public		-
	getNonBMLength	External		-
	getNonBMValues	Public		-
	getTargetNamesAddress	External		-

	getTargetWeightsAddress	External		-
	getBalance	External		-
	getBalanceInUSD	External		-
	getTargetWeight	Public		-
	getUnitPrice	External		-
	getUnitPriceInUSD	External		-
	getLatestPrice	External		-
	getFundValues	External		-
	getTokenValues	External		-
	getFundDataAll	External		-
	totalSupply	Public		-
	_handleFeeTransferSubscribe	Internal	✓	
	_handleFeeTransferRedeem	Internal	✓	
	updateCopyTrading	Public	✓	-
	setCopyTrading	Public	✓	-
	setDCA	Public	✓	-
	setManagerMonthlyFee	External	✓	-
	redeemManagerMonthlyFees	External	✓	-
	setPause	Public	✓	-
	createTargetNames	Public	Payable	-
	updateRebalancePeriod	External	✓	-
	updateManagerProperty	External	Payable	-
	updateNonBMNames	Public	✓	-
	rebalance	Public	Payable	-
	moveNonIndexNameToBase	External	Payable	-
	subscribe	External	Payable	nonReentrant
	redeem	Public	Payable	nonReentrant
	emergencyRedeem	External	Payable	-
	checkSubscription	Public		-
	updatePlatformAddresses	External	✓	-
	updateManagerMaxMonthlyFee	External	✓	-
	_getNonBMLength	Internal		
	_getTargetWeightQty	Internal		
	_getDeleteNames	Internal		
	_getNewFundUnits	Internal		

	_getBalance	Internal		
	_getBalanceInUSD	Internal		
	_getUnitPriceInUSD	Internal		
	_isSmallSubs	Internal		
	_getUnitPrice	Internal		
	_getFundValues	Internal		
	_getTokenValues	Internal		
	_getLatestPrice	Internal		
	_getTransferAmt	Internal		
	_getMintQty	Internal		
	_getActiveOverWeight	Internal		
	_rebalance	Internal	✓	
	_sellOverWeightNames	Internal	✓	
	_buyUnderWeightNames	Internal	✓	
	_moveNonIndexNameToBase	Internal	✓	
	_createTargetNames	Internal	✓	
	_transferNonBM	Internal	✓	
AltairMaster	Implementation	Initializable, OwnableUpgradeable		
	initialize	Public	✓	initializer
	getTokenName	External		-
	changeRouter	Public	✓	onlyAdmin
	getRouterAddress	External		-
	updateTokenNames	External	✓	onlyAdmin
	addPancakePriceToken	Public	✓	onlyAdmin
	getPriceByAllAddress	External		-
	getPriceByAddress	External		-
	getPriceFromBand	External		-
	getQuotes	Public		-
	getPancakePrice	Public		-
	sqrt	Public		-
	getLpPrice	External		-
	use1inchOracle	External		-
	changeChainlinkOracle	Public	✓	onlyAdmin

	useChainlinkOracle	External		-
	updateUseSupportingFeeOnTransferTokens	Public	✓	onlyAdmin
	getUseSupportingFeeOnTransferTokens	External		-
	getPair	External		-
IAltairFactory	Interface			
	altairSwap	External		-
	governor	External		-
	getFunds	External		-
	shop	External		-
	feeManager	External		-
	totalSupply	External		-
	naliAltair	External		-
	monthlyCost	External		-
	subscriptions	External		-
	containsFund	External		-
	getFee	External	✓	-
	getManagerFee	External	✓	-
	treasury	External		-
	fundMaster	External		-
	NATIVE	External		-
	fee	External		-
	subscribeFee	External		-
	redeemFee	External		-
	managerFee	External		-
	rolesContract	External	✓	-
	proxyCallContract	External	✓	-
	swapContract	External		-
	managerOwner	External		-
	getFundId	External		-
	naliToken	External		-
	useSwapInfo	External		-
	mint	External	✓	-
	burn	External	✓	-

	fundsId	External	✓	-
IAltairFund	Interface			
	executeAuthorized	External	✓	-
	setAuthorized	External	✓	-
	setDCA	External	✓	-
	updateManagerMaxMonthlyFee	External	✓	-
	version	External		-
	authorized	External		-
	community	External		-
	dca	External		-
	copytrading	External		-
	sltp	External		-
	executeCustomTx	External	✓	-
	transferWETH	External	Payable	-
	managerMonthlyTimestamp	External		-
	getTargetWeight	External		-
	managerOwner	External		-
	getTargetWeightQty	External		-
	getBalance	External		-
	totalSupply	External		-
	getUnitPrice	External		-
	getUnitPriceInUSD	External		-
	getFundDataAll	External		-
	getFundValues	External		-
	getNonBMLength	External		-
	updateManagerProperty	External	Payable	-
	updateManagerFee	External	Payable	-
	updateRebalancePeriod	External	Payable	-
	redeem	External	Payable	-
	rebalance	External	Payable	-
	subscribe	External	Payable	-
	moveNonIndexNameToBase	External	✓	-
	createTargetNames	External	Payable	-
	emergencyRedeem	External	Payable	-

	getTargetNamesAddress	External		-
	getTargetWeightsAddress	External		-
	updatePlatformAddresses	External	✓	-
	name	External		-
	fund	External		-
	symbol	External		-
	updateCopyTrading	External	✓	-
	rebalanceCycle	External		-
	getTransferAmt	External		-
	nonBMNamesMapping	External		-
	nonBMNamesAddress	External		-
IAltairGovernance	Interface			
	containsPermitted	External		-
IAltairMaster	Interface			
	useChainlinkOracle	External		-
	getPair	External		-
	getLpPrice	External		-
	getTokenName	External		-
	getPriceByAddress	External		-
	getPancakePrice	External		-
	getPriceFromBand	External		-
	getRouterAddress	External		-
	getUseSupportingFeeOnTransferTokens	External		-
IAltairShop	Interface			
	trade	External	✓	-
IAltairSwap	Interface			
	swapBNBToTokens	External	Payable	-
	swapTokenToBNB	External	Payable	-
	addLiquidity	External	Payable	-
	removeLiquidity	External	✓	-

IAltairSwapInfo	Interface			
	getAmountsOut	External		-
	getAmountsIn	External		-
	cBurgerSwapRouter	External		-
	cBakerySwapRouter	External		-
	cPancakeSwapRouter	External		-
	cMSwapMemoryAddr	External		-
HistoricAggregatorInterface	Interface			
	latestAnswer	External		-
	latestTimestamp	External	✓	-
	latestRound	External	✓	-
	getAnswer	External	✓	-
	getTimestamp	External	✓	-
IChainlinkAggregator	Interface	HistoricAggregatorInterface		
	decimals	External	✓	-
	getRoundData	External	✓	-
	latestRoundData	External	✓	-
IContractInitializer	Interface			
	initialize	External	✓	-
IContractInitializerCommunity	Interface			
	initialize	External	✓	-
IDao	Interface			
	name	External		-
	symbol	External		-

	buyGovernanceTokens	External	Payable	-
	burnGovernanceTokens	External	✓	-
	getAllTeammates	External		-
	executePermitted	External	✓	-
	currency	External		-
	isTeammates	External		-
IDaoFactory	Interface			
	monthlyCost	External		-
	subscriptions	External		-
	containsFund	External		-
	subscribe	External	✓	-
IERC1155	Interface	IERC165		
	safeTransferFrom	External	✓	-
	safeBatchTransferFrom	External	✓	-
	balanceOf	External		-
	balanceOfBatch	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
IERC165	Interface			
	supportsInterface	External		-
IERC20	Interface			
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IModuleFactory	Interface			

	containsContract	External		-
INaliAltair	Interface	IERC1155		
	totalSupply	External		-
	mint	External	✓	-
	burn	External	✓	-
	createUri	External		-
	setTokenUri	External	✓	-
	setApprovalForTp	External	✓	-
INaliAltairMini	Interface			
	totalSupply	External		-
	mint	External	✓	-
	burn	External	✓	-
	createUri	External		-
	setTokenUri	External	✓	-
	setApprovalForTp	External	✓	-
INaliToken	Interface			
	totalSupply	External		-
	transferFrom	External	✓	-
	approve	External	✓	-
	balanceOf	External		-
	internalBalanceOf	External		-
	allowance	External		-
IOps	Interface			
	gelato	External		-
	createTaskNoPrepayment	External	✓	-
	cancelTask	External	✓	-
	getTaskIdsByUser	External		-
	getFeeDetails	External		-
	getSelector	External		-
IOracle1Inch	Interface			

	getRate	External		-
IPancakeFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IPancakePair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-

	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IPancakeRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-

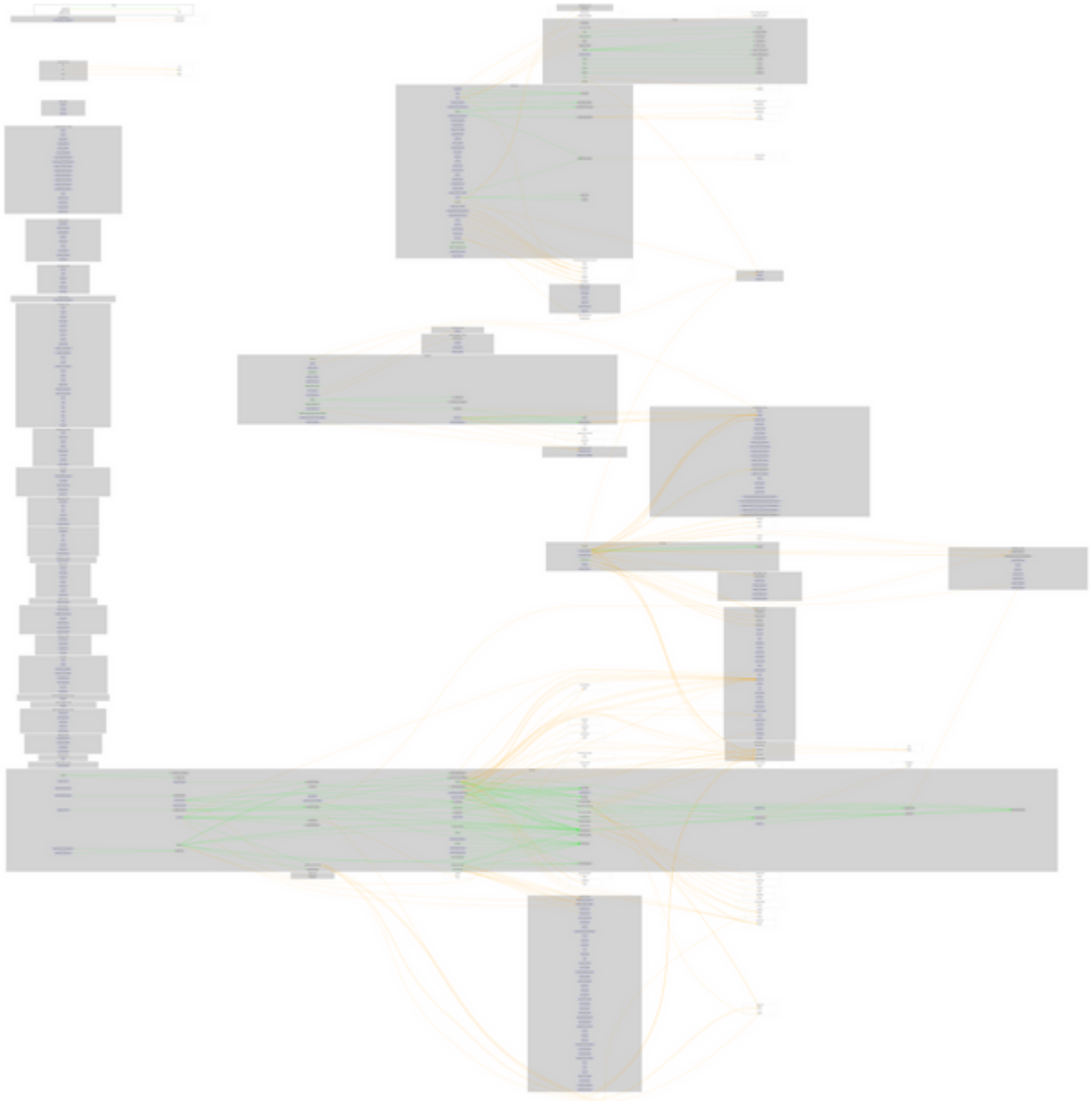
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IProxyCall	Interface			
	proxyCallAndReturnAddress	External	✓	-
IRewardPool	Interface			
	deposit	External	✓	-
	stake	External	✓	-
	withdraw	External	✓	-
	earned	External		-
	getReward	External	✓	-
	balanceOf	External		-
IRoles	Interface			
	isAdmin	External		-
	isOperator	External		-
IStdReference	Interface			
	getReferenceData	External		-
	getReferenceDataBulk	External		-
ITreasury	Interface			
	addLiquidity	External	✓	-
	addStableLiquidity	External	✓	-
	addLiquidityFee	External	✓	-
	buyBack	External	✓	-
	buyBackFee	External	✓	-
	devFee	External	✓	-
	rewardPoolFee	External	✓	-
	rewardPoolAddress	External	✓	-
	devAddress	External	✓	-
IUniswapV2Router01	Interface			

	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IWETH	Interface			
	deposit	External	Payable	-
	transfer	External	✓	-
	withdraw	External	✓	-
AltairLib	Library			
HomoraMath	Library			
	divCeil	Internal		
	fmul	Internal		
	fdiv	Internal		
	sqrt	Internal		
LibrarySwapData	Library			

TransferHelper	Library			
	safeApprove	Internal	✓	
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeTransferBNB	Internal	✓	
AltairSwap	Implementation	Ownable, ReentrancyGuard		
	<Constructor>	Public	✓	-
	configure	External	✓	onlyAdmin
	swapBNBToTokens	External	Payable	-
	swapTokenToBNB	External	Payable	-
	<Receive Ether>	External	Payable	-
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

ExternalProxyCall	Implementation	IProxyCall		
	proxyCallAndReturnAddress	External	✓	-

Contract Flow



Domain Info

Domain Name	nali.finance
Registry Domain ID	c3dcca52c09349afbd0f7b38f3f5cf16-DONUTS
Creation Date	2021-06-08T07:07:24Z
Updated Date	2022-06-06T07:43:05Z
Registry Expiry Date	2023-06-08T07:07:24Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain was created over 1 year before the creation of the audit. It will expire in 6 months.

There is no public billing information, the creator is protected by the privacy settings.

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