Cyberscope

Audit Report

Skate DAO

February 2022

Type	ERC20
Network	ETH
Audited by	© cyberscope

Table of Contents

Contract Review

The audit will review the files that are included in the github repository

Github	https://github.com/artdothaus/skate-dao-contracts
Commit	6935e682ca0c8849de47d0b8b21574fcf6a5291d

Audit Updates

Initial Audit	7th February 2022
Corrected	

Nouns DAO

Nouns is a decentralised autonomous organisation that generates one NFT every day. The users have the ability to make a bid in order to claim the daily NFT. The highest bid will earn the NFT. The remaining bids that have not won are refunded back to the bidders. If there are no bids, the NFT goes to the zero address. The NFT is created by a pseudo-random combination of different traits. The traits are lists of different icons of backgrounds, heads, clothes, etc. The funds from the winning bid are moved to the governor's wallet. The Nouns NFT holders have the ability to create, vote and execute proposals. Some interesting proposals have been executed, like donating 5 ETH to charities.

The nouns organisation contains a lot of bibliography regarding the Nouns ecosystem.

- Quick introduction
- Noun protocol
- Smart Contracts architecture
- Nouns repository

Skate DAO

Skate Dao introduces a novel approach around Nouns Dao. It keeps the fundamental logic of the Nouns Dao but it differentiates on some concepts. In this audit we will focus on the Skate Dao changes over the original Nouns implementation. We will comment on the security, optimization and architectural aspects of the Skate Dao fork. The basic concepts that have been diversified:

Skate DAO	Nouns DAO
Produces a new auction every 666 blocks.	Produces a new action every day.
The bidder chooses to split the amount to charity and the creators.	The bidders amount goes to the creators.
No governor system.	Ruled by a governor system, where Nouns holders can vote and create proposals.
There is no time extension.	The auction's end time is extended if the bid is placed within a certain time frame before the auction's end time.
The auction lifespan can be changed any time from the contract owner.	The auction lifespan cannot be changed. It will always be one day.
The contract owner can withdraw the balance of the auction's contract.	The contract owner cannot withdraw the balance of the auction's contract.

Contract Analysis

Users Claim NFTs Without Auction Participation

Criticality	critical
Location	SkateContract.sol#L121

Description

The users have the ability to mint NFTs to their accounts without participating in the auction process. The users may take this advantage by calling the `settleCurrentAndCreateNewAuction()` function at the end of an auction cycle. The `settleCurrentAndCreateNewAuction()` will settle the current auction and create the next one. During the creation of the next auction, an NFT is minted to the caller's address.

```
_safeMint(msg.sender, tokenId);
```

Recommendation

The new NFTs should be minted to a dedicated entity that will hold the NFT until the end of the auction.

Settled Auction Funds Exploit

Criticality	critical
Location	SkateContract.sol#L157

Description

The winning bidder has the ability to exploit the funds transfer process. The transfer process assumes that the `skatePercent` and the `daoPercent` are summed up to one hundred. During the initiation of a bid, the bidder has the ability to arbitrarily set the value of `skatePercent` and `daoPercent`. For instance, the bidder can set the `skatePercent` and `daoPercent` value to zero. Hence the transfer process can be manipulated.

```
auction.skatePercent = _skatePercent;
auction.daoPercent = _daoPercent;
...
_safeTransferETHWithFallback(skate, _auction.amount * _auction.skatePercent /
100);
_safeTransferETHWithFallback(dao, _auction.amount * _auction.daoPercent / 100);
```

Recommendation

The contract should check that the sum of `skatePercent` and `daoPercent` variables always equals one hundred.

Funds Transfer does not Fallback

Criticality	minor
Location	SkateContract.sol#L269

Description

The `_safeTransferETHWithFallback` implementation does not follow the contract's description. The function is supposed to transfer ETH. If it fails it should wrap the ETH and try to send it as WETH as a fallback. In the actual implementation, the function solely transfers ETH without fallback.

```
/**
* @notice Transfer ETH. If the ETH transfer fails, wrap the ETH and try send it
as WETH.
*/
function _safeTransferETHWithFallback(address to, uint256 amount) internal
returns (bool) {
    (bool success, ) = to.call{ value: amount, gas: 50_000 }(new bytes(0));
    return success;
}
```

Recommendation

The function should either implement a fallback mechanism or change the description and naming.

Contract Owner can Drain the Auction Funds

Criticality	critical
Location	SkateContract.sol#L213

Description

The contract owner has the ability to take all the auction's contract balance. If the function was called during a running auction, then:

- The bidders that lose will not be able to take back their bid amount.
- The winning amount will not be transferred to the Skate and Dao addresses.

```
function withdrawAll() public onlyOwner {
    payable(owner()).transfer(address(this).balance);
}
```

Recommendation

There should be guarantees that the contract owner cannot take the amount that is reserved for the auction's healthy progress.

Contract does not Pause on Failure

Criticality	medium
Location	SkateContract.sol#L169

Description

According to the specification:

- If the auction creation succeeds, there should be an AuctionCreated event.
- If the auction creation fails, the auction should be paused.

The above mentioned specification is not implemented in the contract's function.

```
/**
* @notice Create an auction.
* @dev Store the auction details in the `auction` state variable and emit an
AuctionCreated event.
* If the mint reverts, the minter was updated without pausing this contract
first. To remedy this,
* catch the revert and pause this contract.
*/
function _createAuction() internal {
    uint256 gnarId = mint();
    uint256 startBlock = block.number;
    uint256 endBlock = startBlock + auction_period_blocks;

    auction = Auction({
        gnarId: gnarId,
        amount: 0,
        startBlock: startBlock,
        endBlock: endBlock,
        bidder: payable(0),
        skatePercent: 50,
        daoPercent: 50,
        settled: false
    });
    // emit AuctionCreated(gnarId);
}
```

Recommendation

The contract should implement the described specification.

Sanity check Misused

Criticality	minor
Location	SkateContract.sol#L88

Description

The `_createAuction()` function has some pre-requirements in order to operate properly. These requirements are checked in the `auctionStart()` function but are not checked in the `settleCurrentAndCreateNewAuction()` function. As a result, there is an inconsistency between the pre-requirements.

```
function auctionStart() public onlyOwner {
    require(paused == true, "Auction already started");
    require(skate != address(0), "Skate address must be set");
    require(dao != address(0), "DAO address must be set");
    paused = false;
    _createAuction();
}
```

Recommendation

The sanity check for `skate` and `dao` variables should be checked once in the function that mutates their state (`setSkateDaoAddresses()`).

Duplication of Requirement Check

Criticality	minor
Location	SkateContract.sol#L98

Description

The `settleCurrentAndCreateNewAuction()` requires that the auction should not be settled. This requirement is checked again inside the `_settleAuction()` function. Hence, the outer check is redundant.

```
function settleCurrentAndCreateNewAuction() external nonReentrant {
    require(paused == false, "Auction is paused");
    require(auction.settled == false, "Already settled");
    _settleAuction();
    _createAuction();
}
```

Recommendation

The `auction.settled == false` requirement could be removed since it is redundant.

Workflow inconsistency

Criticality	minor
Location	SkateContract.sol#L233

Description

According to the auction specification, once the current auction is settled, a new auction should be created. In the implementation of the `unpause()` method, if the auction is not settled it one triggers the settle functionality. The creation of the next auction is missing.

```
function unpause() external onlyOwner {
    require(paused == true, "Already Auction running");
    paused = false;
    if (auction.endBlock < block.number){
        if(auction.settled) {
            _createAuction();
        } else {
            _settleAuction();
        }
    }
}
```

Recommendation

The `unpause()` function should follow the auction specification and create the next auction one the current auction is settled.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	L01	Public Function could be Declared External
	L11	Unnecessary Boolean equality
	L07	Missing Events Arithmetic
	L04	Conformance to Solidity Naming Conventions
	L15	Local Scope Variable Shadowing
	L14	Uninitialized Variables in Local Scope

L01 - Public Function could be Declared External

Criticality	minor
Location	SkateContract.sol#L79,86,213,223,274

Description

Public functions that are never called by the contract should be declared external to save gas.

```
remainBlocks
pause
withdrawAll
...
```

Recommendation

Use the external attribute for functions never called from the contract

L11 - Unnecessary Boolean equality

Criticality	minor
Location	SkateContract.sol#L86,96,223,233

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.

```
require(bool,string)(paused == true,Already Auction running)
require(bool,string)(paused == false,Already Paused)
require(bool,string)(auction.settled == false,Already settled)
...
```

Recommendation

Remove the equality to the boolean constant.

L07 - Missing Events Arithmetic

Criticality	minor
Location	SkateContract.sol#L248,256

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
minBidIncrementPercentage = _minBidIncrementPercentage
reservePrice = _reservePrice
```

Recommendation

Emit an event for critical parameter changes.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	GNARDescriptor.sol#L99,109,119,129,139,149 and 22 more SkateContract.sol#L79,106,114,137,248,256 and 4 more

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.

```
auction_period_blocks
_auction_period_blocks
_minBidIncrementPercentage
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	IGNARDescriptor.sol#L43,45,47,49,51,63 and 15 more

Description

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

```
baseURI
glasses
heads
...
```

Recommendation

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

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	libs/MultiPartRLEToSVG.sol#L70,69,72

Description

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

```
part
cursor
buffer
...
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

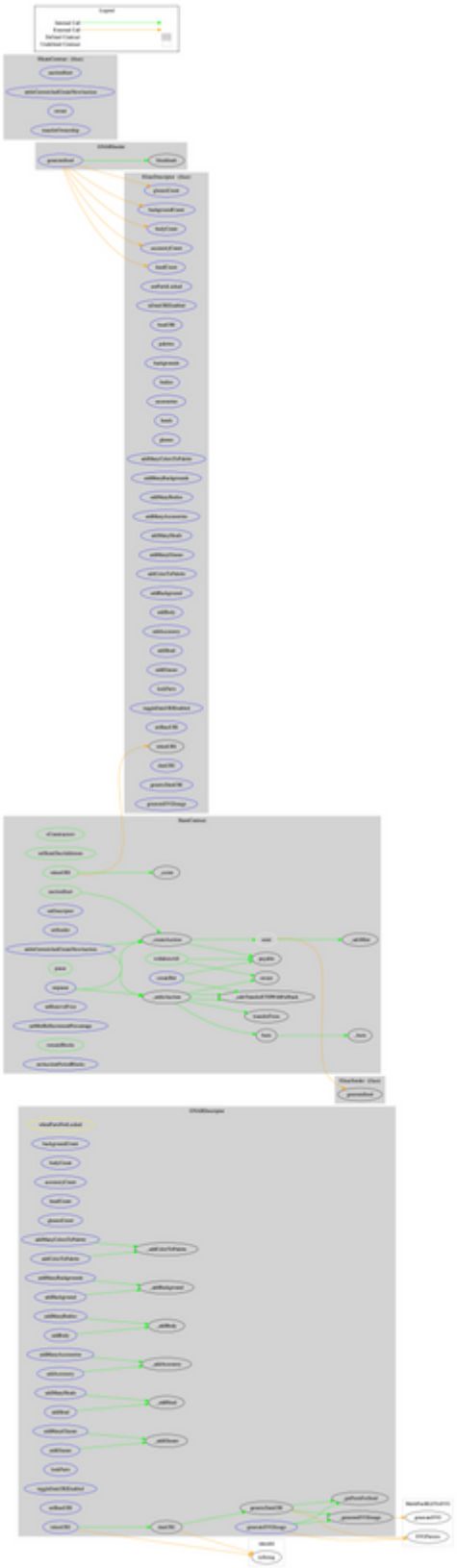
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
GNARDescriptor	Implementation	IGnarDescriptor, Ownable		
	backgroundCount	External		-
	bodyCount	External		-
	accessoryCount	External		-
	headCount	External		-
	glassesCount	External		-
	addManyColorsToPalette	External	✓	onlyOwner
	addManyBackgrounds	External	✓	onlyOwner whenPartsNotLocked
	addManyBodies	External	✓	onlyOwner whenPartsNotLocked
	addManyAccessories	External	✓	onlyOwner whenPartsNotLocked
	addManyHeads	External	✓	onlyOwner whenPartsNotLocked
	addManyGlasses	External	✓	onlyOwner whenPartsNotLocked
	addColorToPalette	External	✓	onlyOwner
	addBackground	External	✓	onlyOwner whenPartsNotLocked
	addBody	External	✓	onlyOwner whenPartsNotLocked
	addAccessory	External	✓	onlyOwner whenPartsNotL

				ocked
	addHead	External	✓	onlyOwner whenPartsNotLocked
	addGlasses	External	✓	onlyOwner whenPartsNotLocked
	lockParts	External	✓	onlyOwner whenPartsNotLocked
	toggleDataURIEnabled	External	✓	onlyOwner
	setBaseURI	External	✓	onlyOwner
	tokenURI	External		-
	dataURI	Public		-
	genericDataURI	Public		-
	generateSVGImage	External		-
	_generateSVGImage	Public		-
	_addColorToPalette	Internal	✓	
	_addBackground	Internal	✓	
	_addBody	Internal	✓	
	_addAccessory	Internal	✓	
	_addHead	Internal	✓	
	_addGlasses	Internal	✓	
	_getPartsForSeed	Internal		
GNARSeeder	Implementation	IGnarSeeder		
	generateSeed	External		-
IGnarDescriptor	Interface			
	arePartsLocked	External	✓	-
	isDataURIEnabled	External	✓	-
	baseURI	External	✓	-
	palettes	External		-
	backgrounds	External		-
	bodies	External		-

	accessories	External		-
	heads	External		-
	glasses	External		-
	backgroundCount	External		-
	bodyCount	External		-
	accessoryCount	External		-
	headCount	External		-
	glassesCount	External		-
	addManyColorsToPalette	External	✓	-
	addManyBackgrounds	External	✓	-
	addManyBodies	External	✓	-
	addManyAccessories	External	✓	-
	addManyHeads	External	✓	-
	addManyGlasses	External	✓	-
	addColorToPalette	External	✓	-
	addBackground	External	✓	-
	addBody	External	✓	-
	addAccessory	External	✓	-
	addHead	External	✓	-
	addGlasses	External	✓	-
	lockParts	External	✓	-
	toggleDataURIEnabled	External	✓	-
	setBaseURI	External	✓	-
	tokenURI	External		-
	dataURI	External		-
	genericDataURI	External		-
	generateSVGImage	External		-
IGnarSeeder	Interface			
	generateSeed	External		-
ISkateContract	Interface	IERC721		
	auctionStart	External	✓	-
	settleCurrentAndCreateNewAuction	External	✓	-

	owner	External	✓	-
	transferOwnership	External	✓	-
SkateContract	Implementation	ERC721Enumerable, Ownable, ReentrancyGuardUpgradeable		
	<Constructor>	Public	✓	ERC721
	setSkateDaoAddresses	Public	✓	onlyOwner
	auctionStart	Public	✓	onlyOwner
	settleCurrentAndCreateNewAuction	External	✓	nonReentrant
	setDescriptor	External	✓	onlyOwner
	setSeeder	External	✓	onlyOwner
	mint	Internal	✓	
	tokenURI	Public		-
	createBid	External	Payable	nonReentrant
	_createAuction	Internal	✓	
	_settleAuction	Internal	✓	
	withdrawAll	Public	✓	onlyOwner
	pause	Public	✓	onlyOwner
	unpause	External	✓	onlyOwner
	setReservePrice	External	✓	onlyOwner
	setMinBidIncrementPercentage	External	✓	onlyOwner
	burn	Public	✓	onlyOwner
	_safeTransferETHWithFallback	Internal	✓	
	remainBlocks	Public		-
	setAuctionPeriodBlocks	External	✓	onlyOwner

Contract Flow



Summary

Skate Dao is a novel addition over the Nouns Dao. It follows the same methodology that is defined on Nouns Dao but it has introduced some diversions.

The main principle of the Nouns Dao is the governance operation. The contract owner for the auctions contract is elected from a timelock system that is controlled by the governance contract. The voters can vote, create and execute proposals. This methodology is missing from the Skate Dao. Hence, the team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Our audit focuses mainly on the changes that have been introduced. We have marked some notes regarding the security aspects, the requirements inconsistency, and performance optimizations.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io