Audit Report

Arc DAO

February 2022

Type	BEP20
Network	BSC TESTNET
Address	0xF1A72607405179109F56e2a7C4F672C5f1090C27
Audited by	© coinscope

Table of Contents

Contract Review

Contract Name	ArcProtocol
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Licence	
Explorer	https://testnet.bscscan.com/token/0xF1A72607405179109F56e2a7C4F672C5f1090C27
Symbol	
Decimals	0
Total Supply	-
Source	contracts/ArcProtocol.sol, contracts/openzeppelin/SafeMath.sol, contracts/openzeppelin/IERC20.sol, contracts/openzeppelin/Initializable.sol, contracts/AggregatorV3Interface.sol, contracts/openzeppelin/AddressUpgradeable.sol
Domain	

Audit Updates

Initial Audit	13th February 2022
Corrected	

Contract Analysis Notes

Price Feed Externa Source

The contract required the latest BNB value in order to determine the maintenance fees amount. Currently, the price feed is working using the binance recommending way via Chainlink Price Feeds according to the oracle principal.

Admin Nodes Creation Permissions

The contract owner has the ability to create new nodes without paying the corresponding fee. They can assign these nodes to any user.

Security Concern

The Arc decentralized autonomous organization provides a lot of functionality to the contract owner. The contract owner has the ability to manipulate the fees, buy nodes without charge and blacklist addresses. Since the contract follows the DAO principals, it could introduce a government pattern similar to the compound protocol in order to limit the power of a single administrator.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	BC	Blacklisted Contracts
●	CO	Code Optimization
●	CR	Code Repetition
●	DSM	Data Structure Misuse
●	MC	Missing Check
●	L01	Public Function could be Declared External
●	L05	Unused State Variable
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination
●	L14	Uninitialized Variables in Local Scope
●	L13	Divide before Multiply Operation

BC - Blacklisted Contracts

Criticality	minor
Location	contract.sol#L505

Description

The contract owner has the authority to stop contracts from transactions. The owner may take advantage of it by calling the `blacklistMalicious` function.

```
function blacklistMalicious(address account, bool value) external {
    _onlyAdmin();
  isBlacklisted[account] = value;
}
```

Recommendation

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L192,222

Description

The expression `nodeTypeId < nodeTypes.length` is redundant since this check is guaranteed during the node creation.

The expression `node.nodeType < nodeTypes.length` is applied after some statements that do not depend on these variables.

```
require(node.nodeType < nodeTypes.length && nodeTypeId < nodeTypes.length, "Not a node type");
```

Recommendation

The expression `node.nodeType < nodeTypes.length` could be removed since it is redundant.

The remaining expression `nodeTypeId < nodeTypes.length` could be moved in the beginning of the function in order to avoid the statements execution in case of a failure.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L192,222

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

```
arcToken.transferFrom(msg.sender, rewardsPool,(nodeTypeTo.price -
nodeTypeFrom.price).mul(rewardPoolFees).div(100));
arcToken.transferFrom(msg.sender, liquidityPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(liquidityPoolFees).div(100));
arcToken.transferFrom(msg.sender, treasuryPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(treasuryFees).div(100));
arcToken.transferFrom(msg.sender, teamMarketingPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(teamMarketingFees).div(100));
```

Recommendation

The expression `nodeTypeTo.price - nodeTypeFrom.price` could be calculated once in a variable.

Criticality	minor
Location	contract.sol#L270,293

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

```
require((msg.value
>(maintenanceFees.mul(10000000000).div(uint256(BNBPrice))).mul(97).div(100)) &&
        (msg.value <
(maintenanceFees.mul(10000000000).div(uint256(BNBPrice))).mul(103).div(100)),
        "Incorrect amount sent to the contract");
```

Recommendation

The expression `maintenanceFees.mul(10000000000).div(uint256(BNBPrice))` could be calculated once in a variable.

DSM - Data Structure Misuse

Criticality	minor
Location	contract.sol#L184,216,247,320,387

Description

The contract uses an ascending indexed array in order to store the nodes for each user. According to the business logic of the contract, a lot of functions are required to find the node structure by the node id. This calculation required the iteration of the array that increased the gas production. The time complexity is O(n).

```
function getNodeIndexWithId(Node[] storage nodes, uint256 id) private view
returns (uint256, bool) {
  for (uint256 i = 0; i < nodes.length; i++) {
    if (nodes[i].id == id) return (i, true);
  }
  return (0, false);
}
```

Recommendation

The contract could use a data structure that provides instant access to the nodes using the id as key.

Proposed structure

```
nodesById: mapping id => node
usersNodes: mapping address => id[]
```

This way the time complexity will be reduced from o(n) to o(1) and the total gas cost will be decreased dramatically.

Proposed replacement

From

```
Node[] storage nodes = usersNodes[msg.sender];
// get the node index
(uint256 nodeIndex, bool finded) = getNodeIndexWithId(nodes, nodeId);
```

To

```
Node node = nodesById[nodeId]
require(node.owner === msg.sender, "Cannot find the node")
```

MC - Missing Check

Criticality	minor
Location	contract.sol#L169,203,234

Description

During the node creation process, the user deposits tokens according to the node price. These tokens are moved proportionally to four pools. The percentages should be summed up to 100 or less. The contract does not contain any check that guarantees the percentage sum-up.

```
    arcToken.transferFrom(msg.sender, rewardsPool,(nodeTypeTo.price -
nodeTypeFrom.price).mul(rewardPoolFees).div(100));
    arcToken.transferFrom(msg.sender, liquidityPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(liquidityPoolFees).div(100));
    arcToken.transferFrom(msg.sender, treasuryPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(treasuryFees).div(100));
    arcToken.transferFrom(msg.sender, teamMarketingPool, (nodeTypeTo.price -
nodeTypeFrom.price).mul(teamMarketingFees).div(100));
```

Recommendation

The contract should guarantee that the sum of the percentage fees will be equal or less to 100. This check should be applied to all the segments that mutate the fees variables.

```
rewardPoolFees + liquidityPoolFees + treasuryFees + teamMarketingFees <= 100
```

L01 - Public Function could be Declared External

Criticality	minor
Location	contracts/ArcProtocol.sol#L73,139,178,212,243,281 and 10 more

Description

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

```
changeMaxNodes
changePoolsFees
changePools
...
```

Recommendation

Use the external attribute for functions never called from the contract

L05 - Unused State Variable

Criticality	minor
Location	contracts/ArcProtocol.sol#L64

Description

There are segments that contain unused state variables.

```
isExcludedFromFees
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/ArcProtocol.sol#L73,513,518,525,533,541 and 8 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.

```
_maxNodes
_liquidityPoolFees
_teamMarketingFees
...
```

Recommendation

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

L09 - Dead Code Elimination

Criticality	minor
Location	contracts/openzeppelin/AddressUpgradeable.sol#L80,90,109,123,142,152 and 3 more contracts/openzeppelin/Initializable.sol#L77 contracts/openzeppelin/SafeMath.sol#L93,191,151,217,168,22 and 4 more

Description

Functions that are not used in the contract, and make the code's size bigger.

```
trySub
tryMul
tryMod
...
```

Recommendation

Remove unused functions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contracts/ArcProtocol.sol#L363,362,364

Description

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

```
allMaintenanceFees
allRewards
allClaimFees
```

Recommendation

All the local scoped variables should be initialized.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/ArcProtocol.sol#L243,281

Description

Performing divisions before multiplications may cause lose of prediction.

```
require(bool,string)((msg.value >
(maintenanceFees.mul(10000000000).div(uint256(BNBPrice))).mul(97).div(100)) &&
(msg.value <
(maintenanceFees.mul(10000000000).div(uint256(BNBPrice))).mul(103).div(100)),Inc
orrect amount sent to the contract)
```

Recommendation

The multiplications should be prior to the divisions.

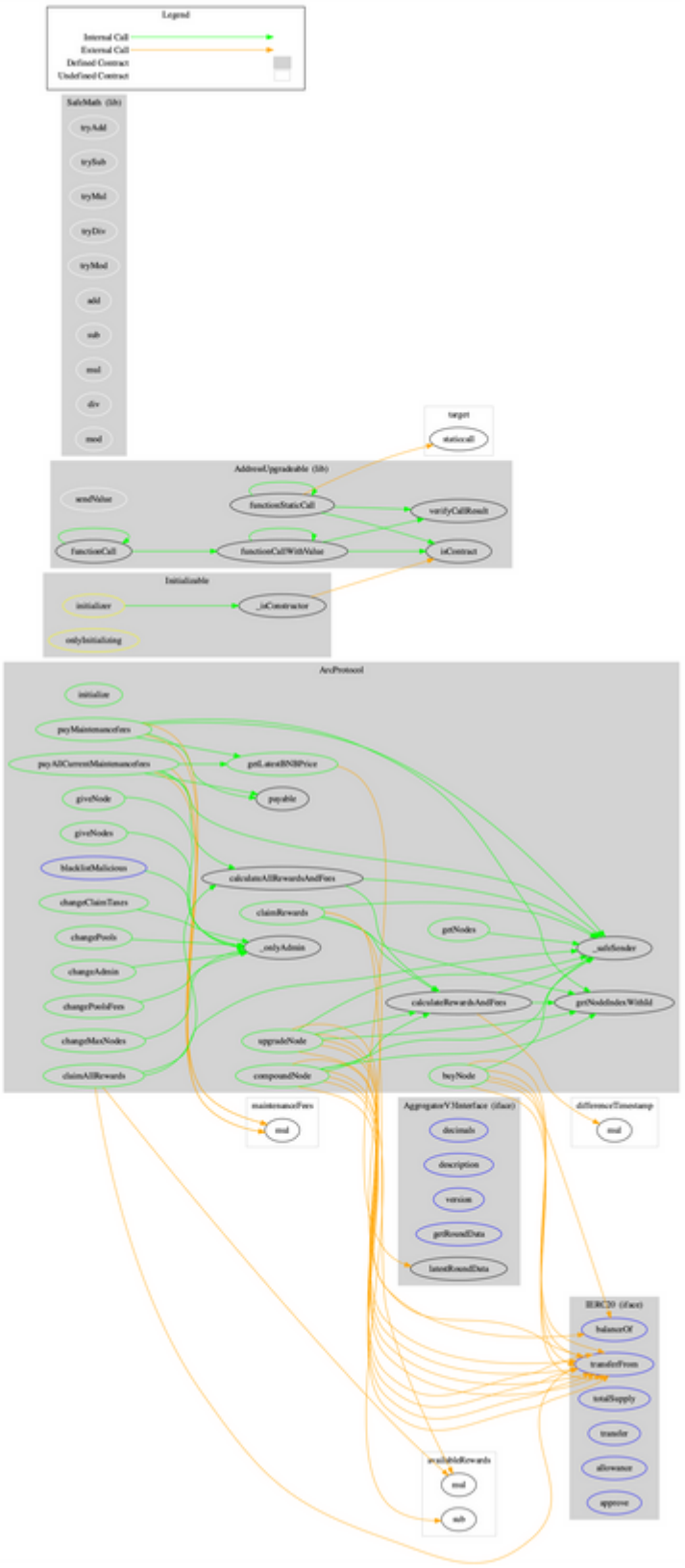
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
AggregatorV3Interface	Interface			
	decimals	External		-
	description	External		-
	version	External		-
	getRoundData	External		-
	latestRoundData	External		-
ArcProtocol	Implementation	Initializable		
	initialize	Public	✓	-
	_onlyAdmin	Private		
	_safeSender	Private		
	getLatestBNBPrice	Public		-
	buyNode	Public	✓	-
	upgradeNode	Public	✓	-
	compoundNode	Public	✓	-
	payMaintenancefees	Public	Payable	-
	payAllCurrentMaintenancefees	Public	Payable	-
	calculateRewardsAndFees	Public		-
	calculateAllRewardsAndFees	Public		-
	claimRewards	Public	✓	-
	claimAllRewards	Public	✓	-
	getNodes	Public		-
	getNodeIndexWithId	Private		
	giveNode	Public	✓	-
	giveNodes	Public	✓	-
	blacklistMalicious	External	✓	-
	changeAdmin	Public	✓	-
	changeClaimTaxes	Public	✓	-

	changePools	Public	✓	-
	changePoolsFees	Public	✓	-
	changeMaxNodes	Public	✓	-
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Initializable	Implementation			
	_isConstructor	Private		
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		

Contract Flow



Summary

Arc DAO is a staking-like mechanism. Users can buy nodes. Each node has a fixed price in Arc tokens. When the users buy a node, the paid amount is moving proportionally to four pools. The node holders have to pay periodically for the “maintenance” cost in BNB. The node holders can claim Arc tokens as a reward.

The audit focuses on security concerns, performance improvements and architectural decisions.

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.

Coinscope 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 Coinscope 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 Coinscope 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 Coinscope team disclaims any liability for the resulting losses.

About Coinscope

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.

Coinscope is aiming to make crypto discoverable and efficient globally. It provides all the essential tools to assist users draw their own conclusions.

The Coinscope.co team

https://www.coinscope.co