Cyberscope

Audit Report

Lava Finance

April 2022

File	LavaFinance.sol
Commit	d59617e3ac107eea6d7601aac6e73e7f45ee00eb
Github	https://github.com/lavafinancial/LavaContracts
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Contract Review 3
Source Files 3
Audit Updates 4
Contract Analysis 5
Admin Privileges 5
CO - Code Optimization (1/2) 6
Description 6
Recommendation 6
Contract Diagnostics 7
CO - Code Optimization (1/2) 8
Description 8
Recommendation 8
CO - Code Optimization (2/2) 9
Description 9
Recommendation 9
MC - Missing Check 10
Description 10
Recommendation 10
L01 - Public Function could be Declared External 11
Description 11
Recommendation 11
L04 - Conformance to Solidity Naming Conventions 12
Description 12
Recommendation 12
L07 - Missing Events Arithmetic 13

Description 13
Recommendation 13
L13 - Divide before Multiply Operation 14
Description 14
Recommendation 14
Unit Test 15
Contract Functions 16
Contract Flow 21
Summary 22
Disclaimer 23
About Cyberscope 24

Contract Review

Github	LavaFinance
commit	d59617e3ac107eea6d7601aac6e73e7f45ee00eb
File	LavaFinance.sol

Source Files

Filename	SHA256
contracts/interfaces/AggregatorV3Interface.sol	398900dac623eff7503ba69036acb204d1650204eb0734eb33b3d7d06c9313d7
contracts/interfaces/GLAVA.sol	c038f9faf68446eec9cff8b4c61b888b526c08608ff9ae85764fd56453c76ed5
contracts/interfaces/IBooster.sol	439b7bd51ee0dfebe347a33396df339746294b1eaf56042aeaabad26dd0a2215
contracts/interfaces/IERC20Burnable.sol	9f760c75e4c4b748b5f137fdc9999dbe38908ef96d113d760ebad335e6c810a0
contracts/interfaces/IFusion.sol	1f4ce7351b4e7a5d185742bdd03c0c7da83bd2c18f1c15376ada25c9c3d2a4b3
contracts/interfaces/IOracle.sol	c0f1901f77769564ef563bf3e8bc49ab86e208800ffbae7af1dc83d4b7d01973
contracts/interfaces/Pair.sol	c976359741ad850af98ccec9bce5fd6d6a2ede9a3d366f5889a245a8c90aab28
contracts/LavaFinance.sol	8a9841cd5124468069019768050ed9212262fd7e50ce397d9feba66e64240cb4

Audit Updates

Initial Audit	9th April 2022
Corrected	

Contract Analysis

The Lava ecosystem is using a lot of contracts As independent entities. This audit focuses on the Finance contract. The Finance contract has the following features:

- The users have the ability to buy nodes according to some predefined tiers.
- A tiers defines the node cost and the rewards that will be distributed.
- The contract is using price oracles to determine the exact cost of the nodes, according to the latest Lava token price.
- Once the user finalizes the bought process, the nodes are minted to the user's address. Additionally, the users receive the corresponding GLava tokens.
- The users receive the rewards proportionally to the timeframe that they have redeemed the latest rewards.
- The users can pay in advance in order to increase the awarded amount.
- The users can choose to receive the rewards in two ways. The first way is to receive lava tokens, the second way is to buy more nodes.
- The users have the ability to upgrade the nodes that they own by using the fuse functionality. Hence, the users can raise their tier by sacrificing the previous nodes if the total cost is the same.

Admin Privileges

- The contract admin has the ability to withdraw all the contract accumulated funds.
- The contract admin has the ability to increase the vest period without limit.
- The contract can set all the trier configuration, the ratios, fees and wallet addresses.

Note

This contract is based on the fact that the GLava contract has mint functionality and the GLava contract has given mint permissions to this contract.

CO - Code Optimization (1/2)

Criticality	minor
Location	contract.sol#L449

Description

The `addMicroNode` method gives the ability to purchase nodes according to the provided amount. If the user provides an amount that is slightly less than the cost of the cheaper tier, then the contract will receive the amount but the user will not receive any token from the transaction.

```
function addMicroNode(address token, uint amount) external nonReentrant {
    require(token == lavaToken || token == pLavaToken, "Only Lava tokens");
    uint totalMicroAmount = amount + microNodes[msg.sender];
    require(totalMicroAmount >= 10 ** lavaDecimals, "Amount too low");
    (, uint minPrice) = getMinNodePrice();
    require(totalMicroAmount <= minPrice + 1e16, "Amount too high");
    IERC20(token).transferFrom(msg.sender, address(this), amount);
    _addMicroNodeAmount(msg.sender, amount);
}
```

Recommendation

The contract should not allow the users to deposit funds if they are not going to receive back the expected tokens.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	CO	Code Optimization
	MC	Missing Check
	L01	Public Function could be Declared External
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L13	Divide before Multiply Operation

CO - Code Optimization (1/2)

Criticality	minor
Location	contract.sol#L449

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

```
function getTokenClaim(address user, uint tokenId, bool useCooldown) public view
returns (uint amount, uint claimTimestamp) {
    require(ownerOf(tokenId) == user, "User Not Owner");
    NodeData storage nd = nodeData[tokenId];
    if (!useCooldown || block.timestamp - nd.lastClaim > claimCooldown) {
        claimTimestamp = nd.paymentExpiry < block.timestamp ? nd.paymentExpiry :
block.timestamp;
        amount = ((claimTimestamp - nd.lastClaim) * tiers[nd.tier].reward) / 1
days;
    }
}
```

Recommendation

The `nd` variable could be declared as *memory* since it is solely used by reading operations.

CO - Code Optimization (2/2)

Criticality	minor
Location	contract.sol#L126,184,485,494

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

```
for (uint i = 1; i < 10; i++) {
```

Recommendation

The contract could use a constant variable for setting the max amount of tiers, rather than repeating the same number.

MC - Missing Check

Criticality	minor
Location	contract.sol#L243,258,286,343,407,411

Description

According to the ERC20 specification, the transfer and transferFrom methods return a boolean that indicates if the transaction succeeded. If the contract ignores this check, then it could guess that the transaction has been accomplished and break the expected business flow.

```
IERC20(token).transferFrom(msg.sender, address(this), cost);
```

Recommendation

The contract should check if the transfer methods have succeeded.

L01 - Public Function could be Declared External

Criticality	minor
Location	contracts/LavaFinance.sol#L94,435,502,513

Description

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

```
setApprovalForAll
getUserNodes
getClaimAmount
initialize
```

Recommendation

Use the external attribute for functions never called from the contract

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/LavaFinance.sol#L94,124,136,140,144,149,154,161,172,176,183,195,200,205,209,418,50,83

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.

```
lavaDecimals
Ratios
_address
_fusionActive
_fusionFees
_claimCooldown
_maxNodeLavaAllowed
_ratio
_pair
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor
Location	contracts/LavaFinance.sol#L140,200,205

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.

```
claimCooldown = _claimCooldown
maxNodeLavaAllowed = _maxNodeLavaAllowed
microReward = _microReward
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/LavaFinance.sol#L228,290,424

Description

Performing divisions before multiplications may cause lose of prediction.

```
nodePriceInUSD = (nodePriceInUsdce * variableAssetPriceInUSD(usdce)) / (10 **
IERC20MetadataUpgradeable(usdce).decimals())
numNodes = totalAmount / tierCost
lavaAmount = (nodePrice * tokenRatio) / 1e4
```

Recommendation

The multiplications should be prior to the divisions.

Unit Test

✓ Test mint restrictions (42ms)

1) Test mint using lava

✓ Test mint using plava (126ms)

✓ Test mint using usdce (107ms)

✓ Test mint using lp (117ms)

✓ Test mint using wavax (135ms)

2) Test micro node

3) Test maintenance fees

4) Test claim

5) Test claim compound

6) Test claim with NFT

7) Test claim with booster

✓ Test node fusion (694ms)

✓ Test node fusion fees (502ms)

✓ Test transfer whitelist (208ms)

✓ Test withdraw

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		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
	_isConstructor	Private		
PausableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Pausable_init	Internal	✓	onlyInitializing
	__Pausable_init_unchained	Internal	✓	onlyInitializing
	paused	Public		-
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
ReentrancyGuardUpgradeable	Implementation	Initializable		
	__ReentrancyGuard_init	Internal	✓	onlyInitializing
	__ReentrancyGuard_init_unchained	Internal	✓	onlyInitializing
IERC20MetadataUpgradeable	Interface	IERC20Upgradeable		

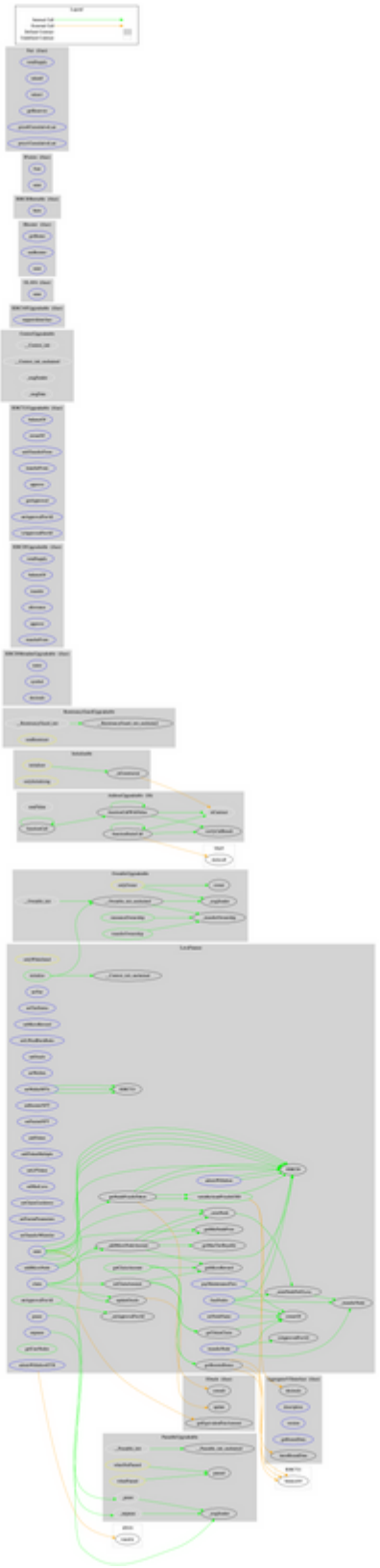
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20Upgradeable	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC721Upgradeable	Interface	IERC165Upgradeable		
	balanceOf	External		-
	ownerOf	External		-
	safeTransferFrom	External	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	getApproved	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		

ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		
IERC165Upgradeable	Interface			
	supportsInterface	External		-
AggregatorV3Interface	Interface			
	decimals	External		-
	description	External		-
	version	External		-
	getRoundData	External		-
	latestRoundData	External		-
GLAVA	Interface			
	mint	External	✓	-
IBooster	Interface			
	getBonus	External		-
	useBooster	External	✓	-
	mint	External	✓	-
IERC20Burnable	Interface			
	burn	External	✓	-
IFusion	Interface			
	fuse	External	✓	-
	mint	External	✓	-
IOracle	Interface			

	update	External	✓	-
	consult	External		-
	getEquivalentPairAmount	External		-
Pair	Interface			
	totalSupply	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
LavaFinance	Implementation	OwnableUpgradeable, PausableUpgradeable, ReentrancyGuardUpgradeable		
	initialize	Public	✓	initializer
	setTier	External	✓	onlyOwner
	setTierStatus	External	✓	onlyOwner
	setMicroReward	External	✓	onlyOwner
	setLPAndBurnRatio	External	✓	onlyOwner
	setOracle	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	setWalletNFTs	External	✓	onlyOwner
	setBoosterNFT	External	✓	onlyOwner
	setFusionNFT	External	✓	onlyOwner
	addToken	External	✓	onlyOwner
	addTokenMultiple	External	✓	onlyOwner
	setLPToken	External	✓	onlyOwner
	setMaxLava	External	✓	onlyOwner
	setClaimCooldown	External	✓	onlyOwner
	setFusionParameters	External	✓	onlyOwner
	setTransferWhitelist	External	✓	onlyOwner
	pause	External	✓	onlyOwner

	unpause	External	✓	onlyOwner
	mint	External	✓	nonReentrant
	_mintNode	Internal	✓	
	_mintNodeNoGLava	Internal	✓	whenNotPaused
	addMicroNode	External	✓	nonReentrant
	_addMicroNodeAmount	Internal	✓	
	payMaintenanceFees	External	✓	nonReentrant
	claim	External	✓	nonReentrant
	setClaimAmount	Internal	✓	
	fuseNodes	External	✓	nonReentrant
	setNodeName	External	✓	-
	updateOracle	Internal	✓	
	adminWithdraw	External	✓	onlyOwner
	adminWithdrawETH	External	✓	onlyOwner
	variableAssetPriceInUSD	Public		-
	getNodePriceInToken	Public		-
	getClaimAmount	Public		-
	getTokenClaim	Public		-
	getMicroReward	Public		-
	getBoosterBonus	Public		-
	getMinNodePrice	Public		-
	getMaxTierBuyable	Public		-
	getUserNodes	Public		-
	ownerOf	Public		-
	setApprovalForAll	Public	✓	-
	_setApprovalForAll	Internal	✓	
	isApprovedForAll	Public		-
	transferNode	External	✓	onlyWhitelisted
	_transferNode	Internal	✓	

Contract Flow



Summary

The Lava Finance contract gives the ability to buy nodes in order to receive rewards in the future. The contract behaves similar to a staking contract with vesting periods. This audit focuses in the business logic, performance improvements, security concerns and potential 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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io