Cyberscope

Audit Report

Lava Finance

April 2022

File	LavaFinance.sol
Commit	605b9971b669eabf3e6727cb61d55f7cdd620e5a
Github	https://github.com/lavafinancial/LavaContracts
Audited by	© cyberscope

Table of Contents

Contract Review

Github	LavaFinance
commit	605b9971b669eabf3e6727cb61d55f7cdd620e5a
File	LavaFinance.sol

Audit Updates

Initial Audit	9th April 2022
Corrected	13th April 2022

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	f0cbb88e6cbc994b565645eabd4320d27d529c7f1f4b3abb5fc263f3961c0a24
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	6e058aaee8c641107b209b62c34d484f2f125a44ecb66f7204a701614dfc1d68
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	8aecaaba0f09bc906c27867246210adfd19230a3e4a209a1909045c633030476
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	b6adbe9bc075b15cfb4b90f1ae020da4c78e3feada056a4c75b875350285c915
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	68bcca423fc72ec9625e219c9e36306c726a347e43f3711467c579bd3f6500c8
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	db1d80b38061ba675444e6ad861a621d99666042950278d6cdeae9a108afdd17

@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol	b7c30218228cde89462d762ff873d1297038bb0f9406da6763bd08e3fccb4f93
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	44edc4d7099c781d11421cea2d82a52948e738f5f6191c8ad01dfc0f9858549c
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol	a39bc026ad6214e9ecd526bd4a1ddf9862d80bd4a9d0d031d9bafa4c3c147c0b
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/interface	8d8c081e548f3f8352218fa545a5b94d01c2221986e73

s/Pair.sol	db1e4e46f7b6d7ef9c1
contracts/LavaFinance.sol	187b59cfc27dbcadabd194e3f6e69ff56cf0713393a82319bc166ea431f83a5f

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

Criticality	minor
Location	contract.sol#L309

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.

Update 13 April

The team has noted that it is not an issue. The amount is added to the flexible node only, which the user is told that it's a permanent deposit.

Contract Diagnostics

Update 13 April

The team has resolved all the issues that were mentioned in the d59617e3ac107eea6d7601aac6e73e7f45ee00eb commit.

Unit Test

✓ Test mint restrictions

✓ Test mint using lava (116ms)

✓ Test mint using plava (117ms)

✓ Test mint using usdce (109ms)

✓ Test mint using lp (109ms)

✓ Test mint using wavax (113ms)

✓ Test micro node (362ms)

✓ Test maintenance fees (181ms)

✓ Test claim (182ms)

✓ Test claim compound (200ms)

✓ Test claim with NFT (140ms)

✓ Test claim with booster (194ms)

✓ Test node fusion (444ms)

✓ Test node fusion fees (226ms)

✓ Test transfer whitelist (155ms)

✓ Test withdraw (40ms)

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		

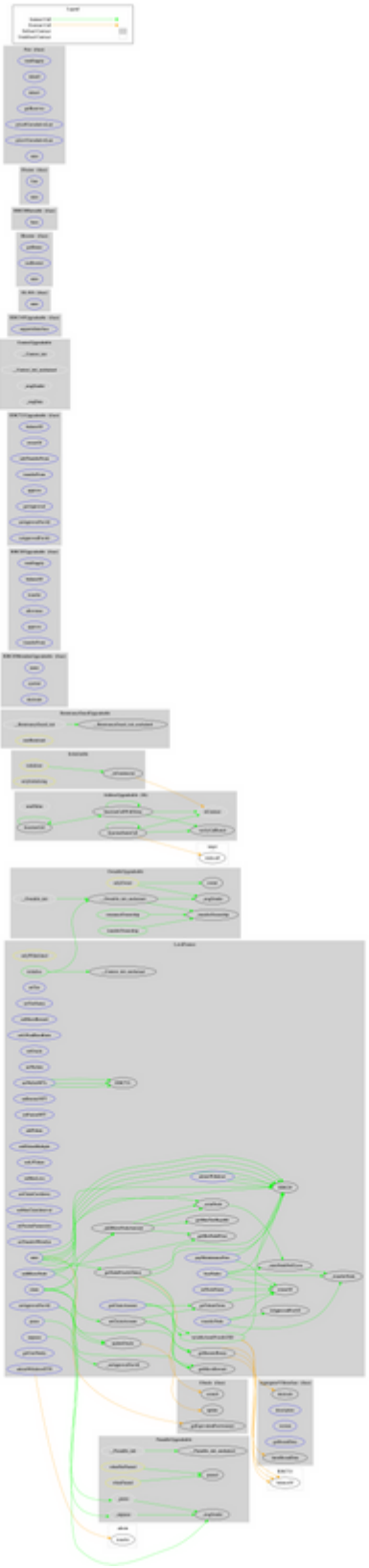
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20Upgradeable	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC721Upgradeable	Interface	IERC165Upgradeable		
	balanceOf	External		-
	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		-
	mint	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
	setMaxClaimInterval	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	External		-
	getTokenClaim	Public		-
	getMicroReward	Internal		
	getBoosterBonus	Public		-
	getMinNodePrice	Public		-
	getMaxTierBuyable	Public		-
	getUserNodes	External		-
	ownerOf	Public		-
	setApprovalForAll	External	✓	-
	_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 on the business logic, performance improvements, security concerns and potential optimizations.

Update 13 April

The team has resolved all the issues that were mentioned in the d59617e3ac107eea6d7601aac6e73e7f45ee00eb commit.

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