Cyberscope

Audit Report

Avalaunch

January 2023

Network https://github.com/avalaunch-external/avalaunch-contracts

Commit b6b8bbb6a326522c07a8cecaa602d8ac30b8d8bc

Audited by © cyberscope

Table of Contents

Review

Contract Name	AvalaunchLBP
Repository	https://github.com/avalaunch-external/avalaunch-contracts
Commit	b6b8bbb6a326522c07a8cecaa602d8ac30b8d8bc

Audit Updates

Initial Audit	09 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	778d5305652c4eb562b12880cb6cf023d67df24844c15783a0b80fac2e715585
@uniswap/lib/contracts/libraries/TransferHelper.sol	22b87fd425d590e533ab7e52478cf72bdc4bde2672e0977c7eff7742e8f0737d
contracts/AvalaunchLBP.sol	91050c5625ddfdd6b593ecbe850c39b5e65411aa7700dda60965bb1f11065a3b
contracts/interfaces/IAsset.sol	f5e80c5fe58e082dea7f43d35bb74959c8d0beed9ab748eccce228fc53ecea64
contracts/interfaces/IJoeFactory.sol	da529dd0580defb1d03899651e1a026a966711b38dccf2c8744af148ef4bda76
contracts/interfaces/IJoeRouter02.sol	740f58a5b6c3edd75b157bd268d346810c27312d7ebb2b97818fcce3a2e9534a
contracts/interfaces/ILBP.sol	b5b5f020d133f0770cf676bac9566154e6e752345630fc835353113c85bea1b6
contracts/interfaces/ILBPFactory.sol	97e092eb8481318e27426b6fe23ac3822e211a97c4bc5d46b401656d53880e82
contracts/interfaces/IVault.sol	0e70065e8a2f2459627063d299cb1441b4753c48e29e57a65075dd3d02442d60

Introduction

AvalaunchLBP manages a Liquidity Bootstrapping Pool (LBP). It creates and registers pools using a LBP factory as an external contract. The contract offers the functionality of finalizing the pools. The finalization can be archived either by transferring the tokens or mounting a share to the liquidity pool.

The scope of this audit is the AvalaunchLBP contract. The audit assumes that the Vault and the LBPFactory are external trusted sources.

Roles

The contract operates as a decentralized autonomous organization (DAO). It does not contain admin role that can configure or mutate the pools and the contract state.

Public

`createLBP()`, Any user can create a LBP by providing the corresponding tokens.

Pool Owner

Pool owner is the addresses that created LBP instances via the `createLBP()` method.

- setSwapEnabled()
- transferPoolOwnership()
- exitPool()
- exitToTraderJoe()

Tokens Transfer

The `safeTransferFrom()` function is used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior. The contract does not implement any mechanism that takes into account the transferred amount reduction. Since the Vault is out of the audit scope, the audit assumes that it tolerates this inconsistency. The `safeApprove()` method could approve the transferred amount rather than the expected amount.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	TII	Token Indexing Improvement	Unresolved
●	MSC	Missing Sanity Check	Unresolved
●	DSM	Data Structure Misuse	Unresolved
●	L05	Unused State Variable	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

TII - Token Indexing Improvement

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L
Status	Unresolved

Description

The contract is using the `isCorrectOrder` boolean indicator to identify the fund and the main token. This is happening because these two tokens are represented by an array of two addresses. The `isCorrectOrder` logic is introduced in the `createLBP()` method. Then, every method is using the `isCorrectOrder` to determine the token that should be accessed. This methodology produces complexity in the source code that decreases the readability and slightly decreases the performance.

```
poolConfig.amounts[poolConfig.isCorrectOrder ? 0 : 1]
uint256 fundTokenIndex = isCorrectOrder ? 0 : 1;
uint256 mainTokenIndex = isCorrectOrder ? 1 : 0;
...
```

Recommendation

The contract could index the tokens when it receives them in the `createLBP()`. So, the contract will know which token is on each index of the array. As a result, the ternary operators will be replaced from direct array indexing.

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L119
Status	Unresolved

Description

The `platformAccessFeeBPS` variable is initialized in the constructor of the contract. It is used to calculate the platform's fee amount. The maximum value that can be set is 10,000 since it is the dominator's value. If the `platformAccessFeeBPS` is initialized with a value greater than 10,000, then the pools will not be able to be finalized.

```
constructor(
    address _lbpFactory,
    uint256 _platformAccessFeeBPS,
    address _traderJoeRouter,
    address _traderJoeFactory
) {
    platformAccessFeeBPS = _platformAccessFeeBPS;
}
```

Recommendation

The team is advised to add a maximum check in the initialization of the `platformAccessFeeBPS` variable.

DSM - Data Structure Misuse

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L55
Status	Unresolved

Description

The contract uses a data structure `_recipientAddresses` to store the platform fee receiver addresses and a data structure `_feeRecipientsBPS` to store the fee of the corresponding receiver. Both variables are initialized once in the constructor and never changed again. The `_distributePlatformAccessFee` method iterates these data structures to proceed with the fee distribution. Since the contract has been implemented to track only one address, the data structures are redundant.

```
mapping(address => uint256) private _feeRecipientsBPS;
EnumerableSet.AddressSet private _recipientAddresses;
```

Recommendation

The team is advised to carefully investigate the usage of the state variables carefully. If a collection is not required, then they can safely replace the data structures with simple types and the repetitive statements with simple ones.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L58
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
uint256 private constant MAX_INT = 2**256 - 1
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L120,127,128
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
lbpFactory = _lbpFactory
traderJoeRouter = _traderJoeRouter
traderJoeFactory = _traderJoeFactory
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/AvalaunchLBP.sol#L2
Status	Unresolved

Description

The ^ symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
EnumerableSet	Library			
	_add	Private	✓	
	_remove	Private	✓	

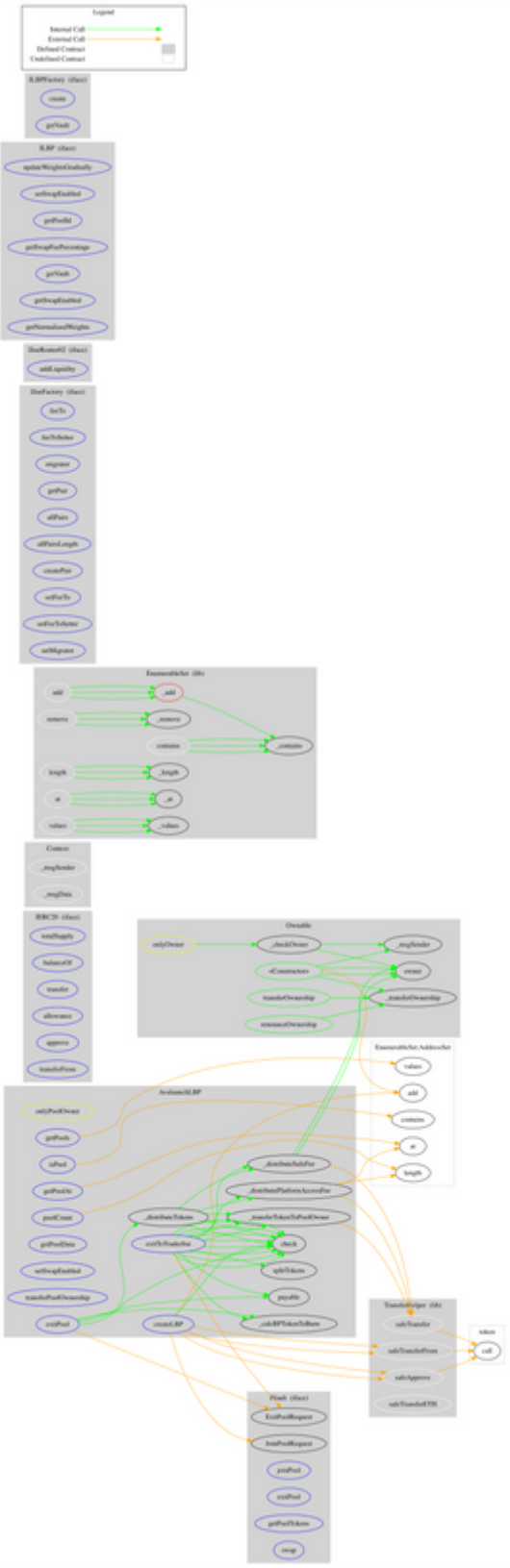
	_contains	Private		
	_length	Private		
	_at	Private		
	_values	Private		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		
TransferHelper	Library			
	safeApprove	Internal	✓	
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeTransferETH	Internal	✓	

AvalaunchLBP	Implementation	Ownable		
		Public	✓	-
	isPool	External		-
	poolCount	External		-
	getPoolAt	External		-
	getPools	External		-
	getPoolData	External		-
	createLBP	External	✓	-
	setSwapEnabled	External	✓	onlyPoolOwner
	transferPoolOwnership	External	✓	onlyPoolOwner
	_calcBPTokenToBurn	Internal		
	splitTokens	Internal		
	exitPool	External	✓	onlyPoolOwner
	exitToTraderJoe	External	✓	onlyPoolOwner
	_distributeTokens	Internal	✓	
	_transferTokenToPoolOwner	Private	✓	
	_distributeSafeFee	Private	✓	
	_distributePlatformAccessFee	Private	✓	
IAsset	Interface			
IJoeFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	migrator	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-

	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
	setMigrator	External	✓	-
IJoeRouter02	Interface			
	addLiquidity	External	✓	-
ILBP	Interface			
	updateWeightsGradually	External	✓	-
	setSwapEnabled	External	✓	-
	getPoolId	External		-
	getSwapFeePercentage	External		-
	getVault	External		-
	getSwapEnabled	External		-
	getNormalizedWeights	External		-
ILBPFactory	Interface			
	create	External	✓	-
	getVault	External	✓	-
IVault	Interface			
	joinPool	External	✓	-
	exitPool	External	✓	-
	getPoolTokens	External		-
	swap	External	Payable	-

Inheritance Graph

Flow Graph



Summary

Avalaunch contract implement a Liquidity Bootstrapping Pool manager mechanism. This audit investigates 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