Damocles

Damocles

Code Security Audit Report

For

DINO

April 2, 2024

Table of Contents

Summary

This report has been prepared for ***DINO*** to discover issues and vulnerabilities in the source code of the ***DINO*** project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following consideration:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

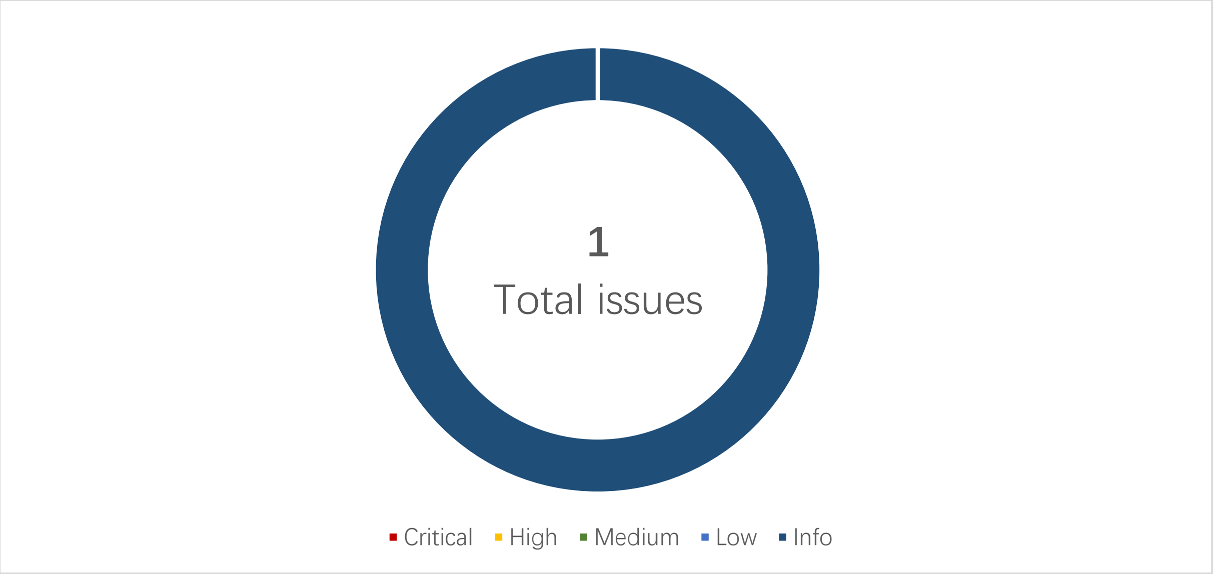
Audit Scope

Contract Name	Lpbexchange/contracts
Platform	Etherum
Language	Solidity
Code Base	https://github.com/WhiteRiverBay/evm-fair-launch
Commit	e82248b9c4fe45ffa2aa6b3ce8784605355a9a86
Address	0x85E90a5430AF45776548ADB82eE4cD9E33B08077

Result Summary

Vulnerability Level	Total	Pending	Solved	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	0	0	0	0
Info	1	1	0	0

Audit Result



DNO-01(INFO): Transferring LP permissions to 0x0 address can provide a higher level of protection for user fairness.

Category	Severity	Location	Status
Optimizable design.	INFO	fair-launch-uniswap-v2.sol:156	**Pending**

Description

Although the current contract does not allow liquidity extraction, if it is refactored and used by other developers, they can still add a function to withdraw liquidity. A fairer approach would be to transfer the liquidity permissions to the 0x0 address, making it impossible for anyone to withdraw the liquidity.

Vulnerability Analysis

None

Recommendation

Transferring LP permissions to 0x0 address.

Reference

Audit Scope

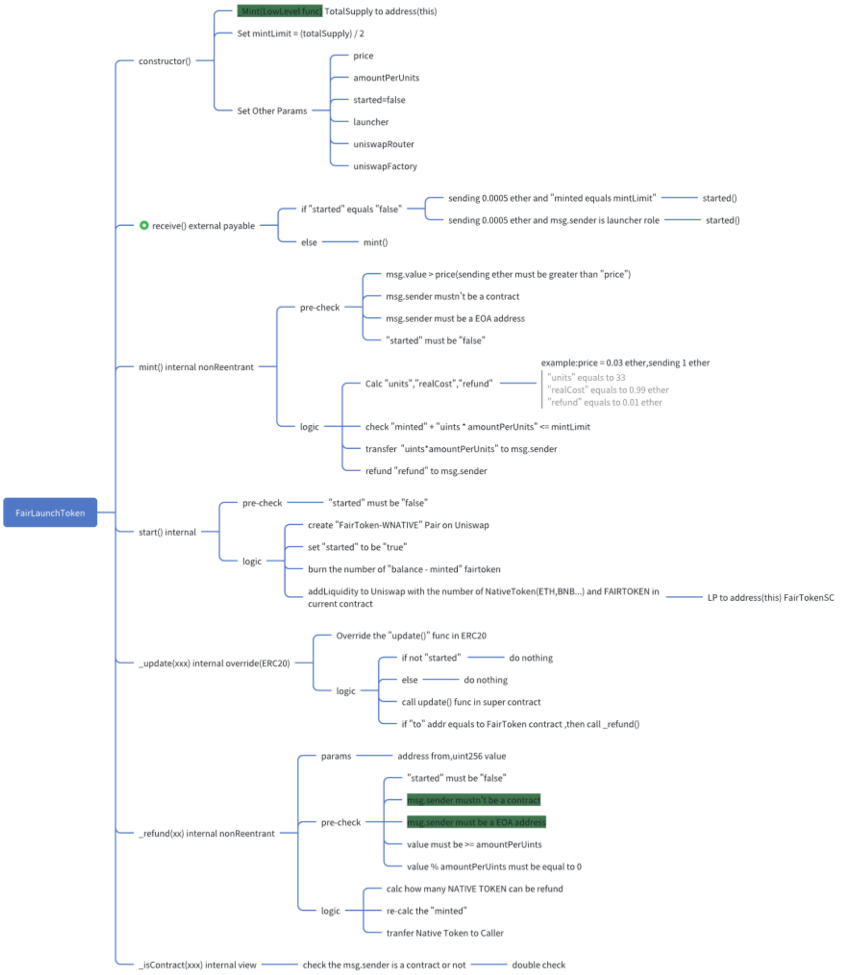
File name	Type	Result	Focus on Issues	Remark
Fair-launch-uniswap-v2	sol	Pass	• Business Logic • Unchecked call return values • Re-entrancy attacks • Denial Of Service attacks • Front Running attacks • Replay signatures attacks • Function default visibility • Loop through long arrays • Wrong inheritance • Unexpected ether balance • Access outside array limits • Delegate calls to untrusted sources • (Regular) calls to untrusted sources • Insecure randomness • Block Timestamp manipulation	Business Logic
Fair-launch-uniswap-v2-with-eth-limit	sol	Pass		Business Logic
ERC20	sol	Pass		standard library
SafeERC20	sol	Pass		standard library
Address	sol	Pass		standard library
Context	sol	Pass		standard library
ReentrancyGuard	sol	Pass		standard library

File name	Type	Result	Focus on Issues	Remark
			• Unupgradable smart contracts • Initializable logic implementations • Different contracts at the same address	

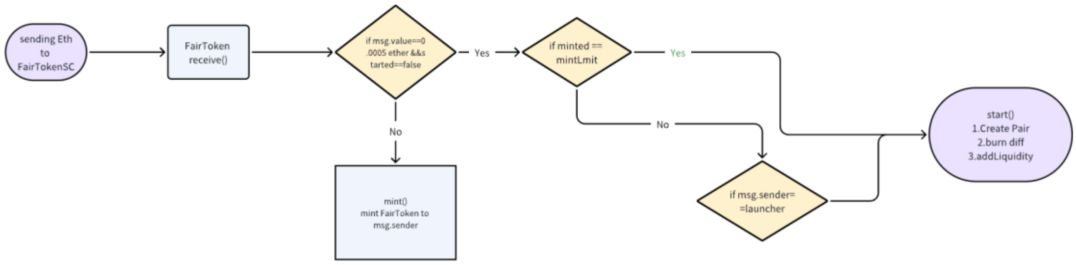
Audit Details

1.Fair-launch-uniswap-v2

Function Detail


FairLaunchToken
constructor()
_Mint(LowLevel func) TotalSupply to address(this)
Set mintLimit = (totalSupply) / 2
Set Other Params
price
amountPerUnits
started=false
launcher
uniswapRouter
uniswapFactory
receive() external payable
if "started" equals "false"
sending 0.0005 ether and "minted equals mintLimit"
started()
sending 0.0005 ether and msg.sender is launcher role
started()
else
mint()
mint() internal nonReentrant
pre-check
msg.value > price(sending ether must be greater than "price")
msg.sender mustn't be a contract
msg.sender must be a EOA address
"started" must be "false"
logic
Calc "units","realCost","refund"
example:price = 0.03 ether,sending 1 ether
"units" equals to 33
"realCost" equals to 0.99 ether
"refund" equals to 0.01 ether
check "minted" + "uints * amountPerUnits" <= mintLimit
transfer "uints*amountPerUnits" to msg.sender
refund "refund" to msg.sender
start() internal
pre-check
"started" must be "false"
logic
create "FairToken-WNATIVE" Pair on Uniswap
set "started" to be "true"
burn the number of "balance - minted" fairtoken
addLiquidity to Uniswap with the number of NativeToken(ETH,BNB...) and FAIRTOKEN in current contract
LP to address(this) FairTokenSC
_update(xxx) internal override(ERC20)
Override the "update()" func in ERC20
logic
if not "started"
do nothing
else
do nothing
call update() func in super contract
if "to" addr equals to FairToken contract ,then call _refund()
_refund(xx) internal nonReentrant
params
address from,uint256 value
pre-check
"started" must be "false"
msg.sender mustn't be a contract
msg.sender must be a EOA address
value must be >= amountPerUints
value % amountPerUints must be equal to 0
logic
calc how many NATIVE TOKEN can be refund
re-calc the "minted"
tranfer Native Token to Caller
_isContract(xxx) internal view
check the msg.sender is a contract or not
double check

Main Process

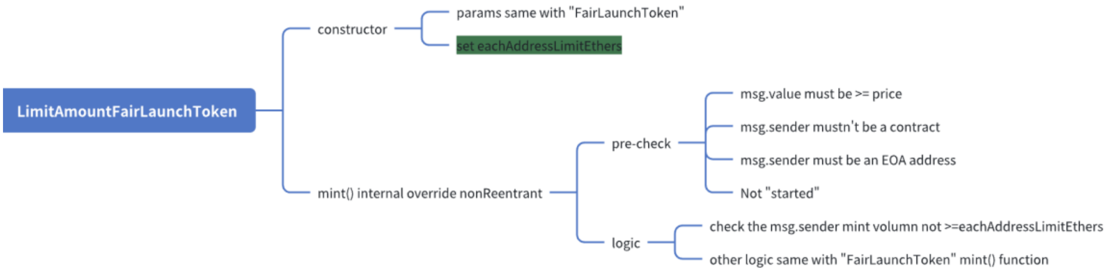

sending Eth to FairTokenSC
FairToken receive()
if msg.value==0 .0005 ether &&s tarted==false
Yes
No
mint() mint FairToken to msg.sender
if minted == mintLmit
Yes
No
if msg.sender= =launcher
start() 1.Create Pair 2.burn diff 3.addLiquidity

The Issues we focused in this audit

Issues	description	YES or Not	remark
Get Token	msg.sender can send "ether" to mint Fairtoken with or without limit (depends on the type of Fair Contract)	YES	mint() function
Sell Token	msg.sender can send Fairtoken to get back "ether"	NO	No such function
addLiquidity	If the "minted" reaches "mintLimit"(totalsupply/2) , any msg.sender can call start() by sending 0.0005 "ether"	YES	start() function
addLiquidity	The launcher role can call start() function	Yes	
removeLiquidity	Someone can remove all the LP tokens on Uniswap	NO（but send LP to 0x0 address is better）	When addLiquidity , the LP token returns to Fairtoken contract,but Fairtoken contract doesn't have the function to transfer LP to anyone.

2.Fair-launch-uniswap-v2-with-eth-limit

Function Detail


LimitAmountFairLaunchToken
constructor
params same with "FairLaunchToken"
set eachAddressLimitEthers
mint() internal override nonReentrant
pre-check
msg.value must be >= price
msg.sender mustn't be a contract
msg.sender must be an EOA address
Not "started"
logic
check the msg.sender mint volumn not >=eachAddressLimitEthers
other logic same with "FairLaunchToken" mint() function

About

Damocles is a 2023 web3 security company specializing in online security services, including smart contract audit, Product audit, penetration testing, GameFi security audit and cheat detection.

Main Web: https://damocleslabs.com/

Medium: https://damocleslabs.medium.com/

Twitter: https://twitter.com/DamoclesLabs

Email: support@damocleslabs.com