SOLIDITYLAB

SMART CONTRACT SECURITY AUDIT OF

Summary

Audit Firm Solidity Lab

Prepared By Chinmay Farkya

Client Firm Raisin Labs

Final Report Date Feb 19th 2023

Audit Summary

Raisin engaged Solidity Lab to review the security of its Smart Contract system. From the 3rd of February to February 19th, a team of 1 auditor reviewed the source code in scope. All findings have been recorded in the following report.

Notice that the examined smart contracts are not resistant to external/internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

✅ Verify the authenticity of this report in Solidity Lab's Portfolio.

Table of Contents

Project Overview

Project Summary

Project Name	Raisin
Language	Solidity
Codebase	https://github.com/crypdoughdoteth/RaisinLabs
Commit	82f43902a93c5076119404865bbfdf00a6f38684

Audit Summary

Delivery Date	February 19th 2023
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	3	3	0	0	0	0
Low	12	12	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum
RS	Raisin.sol	26d8c7c46e12024a3cc3358100a0f589ba0f4bef

Methodology

The auditing process pays special attention to the following considerations:

- 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 aspiring auditors.

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Findings & Resolutions

ID	Title	Category	Severity	Status
RS-1	Missing Governance Functionality	Missing Functionality	Medium	Pending
RS-2	Raiser Can Steal Money When Fund Ends	Logical Error	Medium	Pending
RS-3	Wrong Implementation of calculateFee Operation	Logical Error	Medium	Pending
RS-4	Missing Zero Address Checks	Validation	Low	Pending
RS-5	Wrong Amount Values May Be Used To Initiate A Raisin	Logical Error	Low	Pending
RS-6	Donations Disallowed In The Expiry Block	Logical Error	Low	Pending
RS-7	Trapped ETH	Trapped ETH	Low	Pending
RS-8	Initial Whitelisted Tokens Should Be Documented	Documentation	Low	Pending
RS-9	Funds Can Be Donated Even After The Goal Is Reached	Design	Low	Pending
RS-10	Confusing Event Logic In refund()	Superfluous Code	Low	Pending
RS-11	Incorrect Integration Logic	Documentation	Low	Pending
RS-12	Fees Should Be Refunded When Goal Is Not Reached	Design	Low	Pending
RS-13	Unnecessary Calculations In refund() Function	Superfluous Code	Low	Pending

Findings & Resolutions

ID	Title	Category	Severity	Status
RS-14	State Variable Packing	Optimization	Low	Pending
RS-15	Mark public Functions As external	Optimization	Low	Pending

RS-1 | Missing Governance Functionality

Category	Severity	Location	Status
Missing Functionality	Medium	Raisin.sol	Pending

Description

The whitepaper mentions that there is an address called governance which can end raises abruptly if in some cases it finds it to be malicious. But such a variable has not been declared and not allowed to be calling `endFund()` because of the msg.sender check. At present, any malicious fundraises can not be ended by governance which will impact donors severely.

Recommendation

Add the required functionality, or remove it from the docs.

RS-2 | Raiser Can Steal Money When Fund Ends

Category	Severity	Location	Status
Logical Error	Medium	Raisin.sol: 146-158	Pending

Description

Let's assume that the governance functionality mentioned in M-01 has been set. Now the raiser can call `donateToken` for his own fund whenever governance tries to end his fund citing moderation of suspicious transactions. He will frontrun the `endFund()` function by calling `donateToken()` for any amount shortfall and he can withdraw the funds later by successfully passing the `fundBal` check in `fundWithdraw` function.

The donors will lose all their funds to a malicious raiser.

Recommendation

Add a `msg.sender` check to disallow the raiser of fund index from donating.

RS-3 | Wrong Implementation of calculateFee Operation

Category	Severity	Location	Status
Logical Error	Medium	Raisin.sol: 146-158	Pending

Description

The `calculateFee` instruction at line 153 passes the `msg.sender` of `donateToken` function to check for fees. The `calculateFee` function checks for a partnership with the provided address and allocates fees based on that. This is incorrect because partnership has been set with the raiser of the fund and donor - (`msg.sender` from `donateToken` call) is any random individual who will not have a partnership with raisin.sol. These are 2 different actors.

Recommendation

Replace the parameters in `calculateFee` instruction at Line 153 to `(amount, raisins[index]._raiser)` instead of using `msg.sender`.

RS-4 | Missing Zero Address Checks

Category	Severity	Location	Status
Validation	Low	Raisin.sol: 89-93	Pending

Description

The vault address may be set to the zero address due to admin mistake. While this can be changed later, there is a risk of admin not noticing this.

Recommendation

Add a statement `require(treasury != address(0)` to the line 90.

RS-5 | Wrong Amount Values May Be Used To Initiate A Raisin

Category	Severity	Location	Status
Logical Error	● Low	Raisin.sol: 131-138	Pending

Description

According to the whitepaper, the amount parameter in `initfund()` function represents the minimum % of actual fund required for the fundraising goal to be considered complete. This will not work because there is a check that prevents anything below 100 % to be entered as amount.

Recommendation

The docs should be changed to mention that amount represents the actual fund goal (in wei), since the total goal information is also not being stored.

RS-6 | Donations Disallowed In The Expiry Block

Category	Severity	Location	Status
Logical Error	Low	Raisin.sol: 146-158	Pending

Description

Donations should be only paused for a certain raisin index once the expiry `block.timestamp` has passed. There may be a donator coming into the raisin in the same block in which the fundraise expires and which may complete the goal.

Recommendation

The condition at line 151 should be changed from >= to >. Similarly, the raiser shouldn't be able to withdraw the funds within the expiry block.timestamp. Change condition at line 172 from < to <=.

RS-7 | Trapped ETH

Category	Severity	Location	Status
Trapped ETH	● Low	Raisin.sol: 146-158	Pending

Description

There seems to be no reason to make all of these functions payable. If ETH/other native token on respective chain has to supported as a whitelist token, special logic will be needed because IERC20 doesn't work with ETH. If a user mistakenly sends any ETH with these function calls as they are payable, it will be stuck in the contract forever.

Recommendation

Drop the `payable` mark for all functions.

RS-8 | Initial Whitelisted Tokens Should Be Documented

Category	Severity	Location	Status
Documentation	Low	Raisin.sol	Pending

Description

There are a lot of variations in which ERC20 tokens work. You should document what tokens are going to be used. Also, if ETH isn't to be supported, add a zero address check to `whitelisttoken` function at L#256.

RS-9 | Funds Can Be Donated Even After The Goal Is Reached

Category	Severity	Location	Status
Design	Low	Raisin.sol: 146-158	Pending

Description

It maybe a design choice to let users donate to a fundraise when the goal has reached but it has not expired. I wanted to bring this to your notice since more funds can be collected than required.

RS-10 | Confusing Event Logic In refund()

Category	Severity	Location	Status
Superfluous Code	Low	Raisin.sol: 181-190	Pending

Description

It has been mentioned in the whitepaper at multiple places that events will be used for monitoring by the frontend UI. The refund function emits a `FundEnded` event when the bal = 0 where bal is the `donorBal` of the individual donor. This is incorrect because everytime a donor will refund for a cancelled fund, FundEnded event will be emitted which will mess up onchain monitoring.

Recommendation

Replace bal with fund balance in Line 190.Fund only ends when the entire fund balance goes to zero.

RS-11 | Incorrect Integration Logic

Category	Severity	Location	Status
Documentation	● Low	Raisin.sol	Pending

Description

According to the docs, the integration code functions expect a bool return value from all the functions of raisin, but the actual functions in raisin.sol do not return bool values.

Recommendation

Either the docs are wrong, or the code is wrong. Fix this.

RS-12 | Fees Should Be Refunded When Goal Is Not Reached

Category	Severity	Location	Status
Design	Low	Raisin.sol: 181-190	Pending

Description

If governance address ends mischievous fundraises before goal is reached or the fund expires before completing the goal, the donors receive only the funds they donated minus the fees taken by raisin.sol. In my opinion, fees should only be solidified towards the vault if a fund has completed.

Recommendation

Return fees to donors in these cases.

RS-13 | Unnecessary Calculations In refund() Function

Category	Severity	Location	Status
Superfluous Code	Low	Raisin.sol: 181-190	Pending

Description

The `donorBal[msg.sender][index]` can directly be set to 0 instead of subtracting the bal from the value. The user will be getting the refund in full and only once, so it is better to set it to zero.

Recommendation

Set `donorBal[msg.sender][index]= 0` to save gas.

RS-14 | State Variable Packing

Category	Severity	Location	Status
Optimization	Low	Raisin.sol: 53-57	Pending

Description

Reordering the declaration of these variables to address then uint64 then uint will save one storage slot and thus save deployment gas.

RS-15 | Mark public Functions As external

Category	Severity	Location	Status
Optimization	Low	Raisin.sol: 96-111	Pending

Description

All the get functions from Line 96 to line 111 should be made external instead of public. These functions are not called internally by the contract, so declaring them as external will save gas.

Recommendation

Mark these functions as external.

Disclaimer

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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Solidity Lab to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. Solidity Lab’s position is that each company and individual are responsible for their own due diligence and continuous security. Solidity Lab’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 Solidity Lab is subject to dependencies and 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.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Solidity Lab does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Solidity Lab

Solidity Lab is a community of aspiring auditors guided by Guardian Audits.

To learn more, visit https://lab.guardianaudits.com

To view the Solidity Lab audit portfolio, visit https://github.com/GuardianAudits/LabAudits

To book an audit, message https://t.me/guardianaudits