SOLIDITYLAB

SMART CONTRACT SECURITY AUDIT OF

Summary

Audit Firm Solidity Lab

Prepared By: infamousdegen#8372,0xWeiss#1412 , WangChao#2792

Client Firm Raisin Labs

Final Report Date Feb 19th 2023

Audit Summary

Raisin Labs engaged Solidity Lab to review the security of its Smart Contract system. From the 3rd of February 2023 to 19 February 2023, a team of 3 auditors 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 Labs
Language	Solidity
Codebase	https://github.com/crypdoughdoteth/RaisinLabs/blob/main/src/Raisin.sol
Commit	5b29f55f7982caee0d65ac2e19aa009b024b6c3c

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	1	1	0	0	0	0
High	1	1	0	0	0	0
Medium	1	1	0	0	0	0
Low	2	2	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	Precision Error	Precision Loss	Critical	Pending
RS-2	DOS attack	DoS	High	Pending
RS-3	Use Standard ERC20 Safe Functions	Standard ERC20 Functions	Medium	Pending
RS-4	Minimum tokens requirement	Flexibility	Low	Pending
RS-5	Unnecessary Payable	Unnecessary Payable	Low	Pending

RS-1 | Precision Error

Category	Severity	Location	Status
Precision Loss	Critical	Raisin.sol: 233-236	Pending

Description

For a payment less than 50 the calculate fee will return 0. If the value of the token is high enough this is lost revenue for the protocol.

Recommendation

Refactor the fee logic to avoid precision loss.

RS-2 | DoS Attack

Category	Severity	Location	Status
DoS	High	Raisin.sol: 131	Pending

Description

Create an attacker contract that directly has a loop that creates 2**256-1 funds with 0 amount.

`function initFund (uint amount, IERC20 token, address recipient) external`

This will cause a DoS state and the contract will not be able to create new funds. Having to redeploy the contract.

In Solidity 0.8, if you try to push an item to an array that has already reached its maximum length of 2**256-1, it will revert with an error.

This is because arrays in Solidity have a fixed size and cannot dynamically resize. Attempting to push an item to a full array would result in an overflow, which is disallowed by the language.

Recommendation

Change the array.push method to a mapping. `mapping (uint => Fund) public funds;`

RS-3 | Use Standard ERC20 Safe Functions

Category	Severity	Location	Status
Standard ERC20 Functions	Medium	Raisin.sol: 214-229	Pending

Description

Better to use the OpenZeppelin `safeTransferFrom` instead of manual check. Immediately we don't see any issue but for a critical function like `tokenTransferFrom` following the standard library OpenZeppelin's `safeTransferFrom` is better.

Recommendation

Use the standard ERC20 safe methods provided by OpenZeppelin.

RS-4 | Minimum Token Requirements

Category	Severity	Location	Status
Flexibility	Low	Raisin.sol: 136	Resolved

Description

For tokens like `weth` requiring minimum 100 `weth` is very large amount .This limits the options available for the borrower to raise tokens.

Recommendation

Consider lowering or removing this requirement.

RS-5 | Unnecessary Payable

Category	Severity	Location	Status
Unnecessary Payable	Low	Raisin.sol: 194-195	Resolved

Description

The `payable` specifier is not needed. As the function won't receive ether.

Recommendation

Remove the `payable` specifier.

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