Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Rocket Pool ETH
Website:	rocketpool.net
Platform:	Ethereum
Language:	Solidity
Date:	April 24th, 2024

Table of contents

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the smart contracts of Rocket Pool ETH Token from rocketpool.net were audited. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 24th, 2024.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- RocketTokenRETH is a tokenized stake in the Rocket Pool network, backed by ETH at a variable exchange rate subject to liquidity.
- Rocket Pool ETH Token smart contracts offer various functions like minting, burning, depositing excess, withdrawing deposit collateral, and depositing excess collateral.
- This contract facilitates the minting and burning of rETH tokens, which represent tokenized stakes in the Rocket Pool network, backed by ETH. It ensures the exchangeability of rETH tokens with ETH at variable exchange rates. Additionally, it handles the deposit and withdrawal of ETH collateral from the Rocket Pool deposit pool.

Audit scope

Name	**Code Review and Security Analysis Report for Rocket Pool ETH Smart Contract**
Platform	**Ethereum**
File	RocketTokenRETH.sol
Ethereum Code	0xae78736cd615f374d3085123a210448e74fc6393
Audit Date	April 24th, 2024

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Rocket Pool ETH • Symbol: rETH • Decimals: 18 • Version: 1	**YES, This is valid.**
RocketDepositPool contract control: • Deposit excess ETH from the deposit pool. • Mint a rETH.	**YES, This is valid.**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.

We used various tools like Slither, Solhint, and Remix IDE. At the same time, this finding is based on a critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit Overview section. A general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 0 low, and 3 very low level issues.

Investor Advice: A technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner-controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Moderated
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Business Risk Analysis

Category	Result
Buy Tax	0%
Sell Tax	0%
Cannot Buy	No
Cannot Sell	No
Max Tax	0%
Modify Tax	Not Detected
Fee Check	No
Is Honeypot	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	No
Pause Transfer?	No
Max Tax?	No
Is it Anti-whale?	No
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	No
Can Mint?	Yes
Is it a Proxy?	No
Can Take Ownership?	No
Hidden Owner?	No
Self Destruction?	No
Auditor Confidence	High

Overall Audit Result: **PASSED**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract. Smart contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in Rocket Pool ETH are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties/methods can be reused many times by other contracts in the Rocket Pool ETH.

The EtherAuthority team has no scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a Rocket Pool ETH Token smart contract code in the form of an Etherscan web link.

As mentioned above, code parts are well commented on. and the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries used in this smart contract infrastructure are based on well-known industry standard open-source projects.

Apart from libraries, its functions are not used in external smart contract calls.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyLatestNetworkContract	modifier	Passed	No Issue
3	onlyLatestContract	modifier	Passed	No Issue
4	onlyRegisteredNode	modifier	Passed	No Issue
5	onlyTrustedNode	modifier	Passed	No Issue
6	onlyRegisteredMinipool	modifier	Passed	No Issue
7	onlyGuardian	modifier	Passed	No Issue
8	getContractAddress	internal	Passed	No Issue
9	getContractAddressUnsafe	internal	Passed	No Issue
10	getContractName	internal	Passed	No Issue
11	getRevertMsg	internal	Passed	No Issue
12	getAddress	internal	Passed	No Issue
13	getUint	internal	Passed	No Issue
14	getString	internal	Passed	No Issue
15	getBytes	internal	Passed	No Issue
16	getBool	internal	Passed	No Issue
17	getInt	internal	Passed	No Issue
18	getBytes32	internal	Passed	No Issue
19	setAddress	internal	Passed	No Issue
20	setUint	internal	Passed	No Issue
21	setString	internal	Passed	No Issue
22	setBytes	internal	Passed	No Issue
23	setBool	internal	Passed	No Issue
24	setInt	internal	Passed	No Issue
25	setBytes32	internal	Passed	No Issue
26	deleteAddress	internal	Passed	No Issue
27	deleteUint	internal	Passed	No Issue
28	deleteString	internal	Passed	No Issue
29	deleteBytes	internal	Passed	No Issue
30	deleteBool	internal	Passed	No Issue
31	deleteInt	internal	Passed	No Issue
32	deleteBytes32	internal	Passed	No Issue
33	addUint	internal	Passed	No Issue
34	subUint	internal	Passed	No Issue
35	name	read	Passed	No Issue
36	symbol	read	Passed	No Issue
37	decimals	read	Passed	No Issue
38	totalSupply	read	Passed	No Issue
39	balanceOf	read	Passed	No Issue
40	transfer	write	Passed	No Issue
41	allowance	read	Passed	No Issue
42	approve	write	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

43	transferFrom	write	Passed	No Issue
44	increaseAllowance	write	Passed	No Issue
45	decreaseAllowance	write	Passed	No Issue
46	_transfer	internal	Passed	No Issue
47	_mint	internal	Passed	No Issue
48	_burn	internal	Passed	No Issue
49	_approve	internal	Passed	No Issue
50	_setupDecimals	internal	Passed	No Issue
51	_beforeTokenTransfer	internal	Passed	No Issue
52	receive	external	Passed	No Issue
53	getEthValue	read	Passed	No Issue
54	getRethValue	read	Passed	No Issue
55	getExchangeRate	external	Passed	No Issue
56	getTotalCollateral	read	Passed	No Issue
57	getCollateralRate	read	Passed	No Issue
58	depositExcess	external	Centralized Ownership and Privileges Management	Refer Audit Findings
59	mint	external	Centralized Ownership and Privileges Management	Refer Audit Findings
60	burn	external	Passed	No Issue
61	withdrawDepositCollateral	write	Passed	No Issue
62	depositExcessCollateral	external	Passed	No Issue
63	_beforeTokenTransfer	internal	Passed	No Issue

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Consistent Pragma Solidity Version Usage: **RocketTokenRETH.sol**

```
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['0.7.6', '>=0.6.0<0.8.0']
        - 0.7.6 (RocketBase.sol#27)
        - 0.7.6 (RocketDAOProtocolSettingsNetworkInterface.sol#27)
        - 0.7.6 (RocketDepositPoolInterface.sol#27)
        - 0.7.6 (RocketNetworkBalancesInterface.sol#27)
        - 0.7.6 (RocketStorageInterface.sol#27)
        - 0.7.6 (RocketTokenRETH.sol#27)
        - 0.7.6 (RocketTokenRETHInterface.sol#27)
        - >=0.6.0<0.8.0 (Context.sol#3)
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - >=0.6.0<0.8.0 (IERC20.sol#3)
        - >=0.6.0<0.8.0 (SafeMath.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

Detected different Solidity versions are used.

Resolution: We suggest using one Solidity version.

(2) Centralized Ownership and Privileges Management: **RocketTokenRETH.sol**

The onlyLatestContract as an owner of these functions:

- depositExcess

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

- mint

Resolution: We suggest carefully managing the owner of these functions.

(3) Use the Latest Solidity Compiler Version for Enhanced Security:

RocketTokenRETH.sol

```
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['0.7.6', '>=0.6.0<0.8.0']
        - 0.7.6 (RocketBase.sol#27)
        - 0.7.6 (RocketDAOProtocolSettingsNetworkInterface.sol#27)
        - 0.7.6 (RocketDepositPoolInterface.sol#27)
        - 0.7.6 (RocketNetworkBalancesInterface.sol#27)
        - 0.7.6 (RocketStorageInterface.sol#27)
        - 0.7.6 (RocketTokenRETH.sol#27)
        - 0.7.6 (RocketTokenRETHInterface.sol#27)
        - >=0.6.0<0.8.0 (Context.sol#3)
        - >=0.6.0<0.8.0 (ERC20.sol#3)
        - >=0.6.0<0.8.0 (IERC20.sol#3)
        - >=0.6.0<0.8.0 (SafeMath.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

The solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

Resolution: Deploy with any of the following Solidity versions:

0.8.18

The recommendations take into account:

- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

StakedAaveV3.sol

- mint: Mint rETH only accepts calls from the RocketDepositPool contract.
- depositExcess: Deposit excess ETH from the deposit pool only accepts calls from the RocketDepositPool contract.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Conclusion

We were given a contract code in the form of Etherscan web links. And we have used all possible tests based on given objects as files. We had observed 3 informational issues in the smart contracts. And those issues are not critical. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of the systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and white box penetration testing. We look at the project's website to get a high-level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

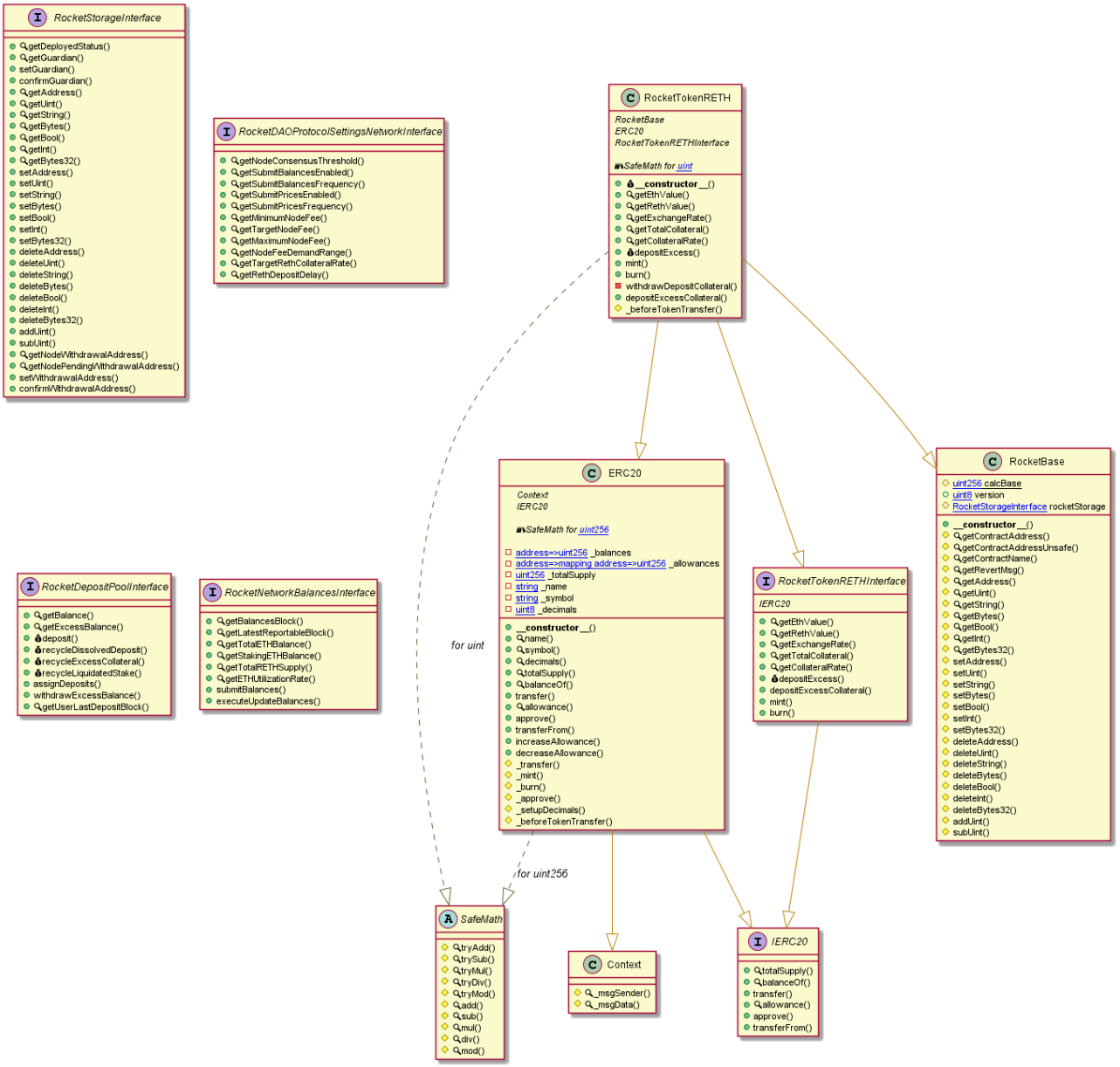
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Appendix

Code Flow Diagram - Rocket Pool ETH



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither Log >> RocketTokenRETH.sol

```
Reentrancy in RocketTokenRETH.burn(uint256) (RocketTokenRETH.sol#961-978):
        External calls:
        - _burn(msg.sender,_rethAmount) (RocketTokenRETH.sol#971)
                - rocketStorage.deleteUint(_key) (RocketTokenRETH.sol#830)
        - withdrawDepositCollateral(ethAmount) (RocketTokenRETH.sol#973)
                - rocketDepositPool.withdrawExcessBalance(_ethRequired.sub(ethBalance)) (RocketTokenRETH.sol#987)
        External calls sending eth:
        - msg.sender.transfer(ethAmount) (RocketTokenRETH.sol#975)
        Event emitted after the call(s):
        - TokensBurned(msg.sender,_rethAmount,ethAmount,block.timestamp) (RocketTokenRETH.sol#977)
Reentrancy in RocketTokenRETH.mint(uint256,address) (RocketTokenRETH.sol#949-958):
        External calls:
        - _mint(_to,rethAmount) (RocketTokenRETH.sol#955)
                - rocketStorage.deleteUint(_key) (RocketTokenRETH.sol#830)
        Event emitted after the call(s):
        - TokensMinted(_to,rethAmount,_ethAmount,block.timestamp) (RocketTokenRETH.sol#957)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

RocketBase.getRevertMsg(bytes) (RocketTokenRETH.sol#794-802) uses assembly
        - INLINE ASM (RocketTokenRETH.sol#797-800)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Context._msgData() (RocketTokenRETH.sol#403-406) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (RocketTokenRETH.sol#681) is never used and should be removed
ERC20._setupDecimals(uint8) (RocketTokenRETH.sol#663-665) is never used and should be removed
RocketBase.addUint(bytes32,uint256) (RocketTokenRETH.sol#838) is never used and should be removed
RocketBase.deleteAddress(bytes32) (RocketTokenRETH.sol#829) is never used and should be removed
RocketBase.deleteBool(bytes32) (RocketTokenRETH.sol#833) is never used and should be removed
RocketBase.deleteBytes(bytes32) (RocketTokenRETH.sol#832) is never used and should be removed
RocketBase.deleteBytes32(bytes32) (RocketTokenRETH.sol#835) is never used and should be removed
RocketBase.deleteInt(bytes32) (RocketTokenRETH.sol#834) is never used and should be removed
RocketBase.deleteString(bytes32) (RocketTokenRETH.sol#831) is never used and should be removed
RocketBase.getBool(bytes32) (RocketTokenRETH.sol#815) is never used and should be removed
RocketBase.getBytes(bytes32) (RocketTokenRETH.sol#814) is never used and should be removed
RocketBase.getBytes32(bytes32) (RocketTokenRETH.sol#817) is never used and should be removed
RocketBase.getContractAddressUnsafe(string) (RocketTokenRETH.sol#775-780) is never used and should be removed
RocketBase.getContractName(address) (RocketTokenRETH.sol#784-791) is never used and should be removed
RocketBase.getInt(bytes32) (RocketTokenRETH.sol#816) is never used and should be removed
RocketBase.getRevertMsg(bytes) (RocketTokenRETH.sol#794-802) is never used and should be removed
RocketBase.getString(bytes32) (RocketTokenRETH.sol#813) is never used and should be removed
RocketBase.setAddress(bytes32,address) (RocketTokenRETH.sol#820) is never used and should be removed
RocketBase.setBool(bytes32,bool) (RocketTokenRETH.sol#824) is never used and should be removed
RocketBase.setBytes(bytes32,bytes) (RocketTokenRETH.sol#823) is never used and should be removed
RocketBase.setBytes32(bytes32,bytes32) (RocketTokenRETH.sol#826) is never used and should be removed
RocketBase.setInt(bytes32,int256) (RocketTokenRETH.sol#825) is never used and should be removed
RocketBase.setString(bytes32,string) (RocketTokenRETH.sol#822) is never used and should be removed
RocketBase.setUint(bytes32,uint256) (RocketTokenRETH.sol#821) is never used and should be removed
RocketBase.subUint(bytes32,uint256) (RocketTokenRETH.sol#839) is never used and should be removed
SafeMath.div(uint256,uint256,string) (RocketTokenRETH.sol#264-267) is never used and should be removed
SafeMath.mod(uint256,uint256) (RocketTokenRETH.sol#226-229) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (RocketTokenRETH.sol#284-287) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (RocketTokenRETH.sol#98-102) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (RocketTokenRETH.sol#134-137) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (RocketTokenRETH.sol#144-147) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (RocketTokenRETH.sol#119-127) is never used and should be removed
SafeMath.trySub(uint256,uint256) (RocketTokenRETH.sol#109-112) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Pragma version>=0.6.0<0.8.0 (RocketTokenRETH.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
Parameter RocketBase.getContractAddress(string)._contractName (RocketTokenRETH.sol#764) is not in mixedCase
Parameter RocketBase.getContractAddressUnsafe(string)._contractName (RocketTokenRETH.sol#775) is not in mixedCase
Parameter RocketBase.getContractName(address)._contractAddress (RocketTokenRETH.sol#784) is not in mixedCase
Parameter RocketBase.getRevertMsg(bytes)._returnData (RocketTokenRETH.sol#794) is not in mixedCase
Parameter RocketBase.getAddress(bytes32)._key (RocketTokenRETH.sol#811) is not in mixedCase
Parameter RocketBase.getUint(bytes32)._key (RocketTokenRETH.sol#812) is not in mixedCase
Parameter RocketBase.getString(bytes32)._key (RocketTokenRETH.sol#813) is not in mixedCase
Parameter RocketBase.getBytes(bytes32)._key (RocketTokenRETH.sol#814) is not in mixedCase
Parameter RocketBase.getBool(bytes32)._key (RocketTokenRETH.sol#815) is not in mixedCase
Parameter RocketBase.getInt(bytes32)._key (RocketTokenRETH.sol#816) is not in mixedCase
Parameter RocketBase.getBytes32(bytes32)._key (RocketTokenRETH.sol#817) is not in mixedCase
Parameter RocketBase.setAddress(bytes32,address)._key (RocketTokenRETH.sol#820) is not in mixedCase
Parameter RocketBase.setAddress(bytes32,address)._value (RocketTokenRETH.sol#820) is not in mixedCase
Parameter RocketBase.setUint(bytes32,uint256)._key (RocketTokenRETH.sol#821) is not in mixedCase
Parameter RocketBase.setUint(bytes32,uint256)._value (RocketTokenRETH.sol#821) is not in mixedCase
Parameter RocketBase.setString(bytes32,string)._key (RocketTokenRETH.sol#822) is not in mixedCase
Parameter RocketBase.setString(bytes32,string)._value (RocketTokenRETH.sol#822) is not in mixedCase
Parameter RocketBase.setBytes(bytes32,bytes)._key (RocketTokenRETH.sol#823) is not in mixedCase
Parameter RocketBase.setBytes(bytes32,bytes)._value (RocketTokenRETH.sol#823) is not in mixedCase
Parameter RocketBase.setBool(bytes32,bool)._key (RocketTokenRETH.sol#824) is not in mixedCase
Parameter RocketBase.setBool(bytes32,bool)._value (RocketTokenRETH.sol#824) is not in mixedCase
Parameter RocketBase.setInt(bytes32,int256)._key (RocketTokenRETH.sol#825) is not in mixedCase
Parameter RocketBase.setInt(bytes32,int256)._value (RocketTokenRETH.sol#825) is not in mixedCase
Parameter RocketBase.setBytes32(bytes32,bytes32)._key (RocketTokenRETH.sol#826) is not in mixedCase
Parameter RocketBase.setBytes32(bytes32,bytes32)._value (RocketTokenRETH.sol#826) is not in mixedCase
Parameter RocketBase.deleteAddress(bytes32)._key (RocketTokenRETH.sol#829) is not in mixedCase
Parameter RocketBase.deleteUint(bytes32)._key (RocketTokenRETH.sol#830) is not in mixedCase
Parameter RocketBase.deleteString(bytes32)._key (RocketTokenRETH.sol#831) is not in mixedCase
Parameter RocketBase.deleteBytes(bytes32)._key (RocketTokenRETH.sol#832) is not in mixedCase
Parameter RocketBase.deleteBool(bytes32)._key (RocketTokenRETH.sol#833) is not in mixedCase
Parameter RocketBase.deleteInt(bytes32)._key (RocketTokenRETH.sol#834) is not in mixedCase
Parameter RocketBase.deleteBytes32(bytes32)._key (RocketTokenRETH.sol#835) is not in mixedCase
Parameter RocketBase.addUint(bytes32,uint256)._key (RocketTokenRETH.sol#838) is not in mixedCase
Parameter RocketBase.addUint(bytes32,uint256)._amount (RocketTokenRETH.sol#838) is not in mixedCase
Parameter RocketBase.subUint(bytes32,uint256)._key (RocketTokenRETH.sol#839) is not in mixedCase
Parameter RocketBase.subUint(bytes32,uint256)._amount (RocketTokenRETH.sol#839) is not in mixedCase
```

```
Parameter RocketBase.deleteInt(bytes32)._key (RocketTokenRETH.sol#834) is not in mixedCase
Parameter RocketBase.deleteBytes32(bytes32)._key (RocketTokenRETH.sol#835) is not in mixedCase
Parameter RocketBase.addUint(bytes32,uint256)._key (RocketTokenRETH.sol#838) is not in mixedCase
Parameter RocketBase.addUint(bytes32,uint256)._amount (RocketTokenRETH.sol#838) is not in mixedCase
Parameter RocketBase.subUint(bytes32,uint256)._key (RocketTokenRETH.sol#839) is not in mixedCase
Parameter RocketBase.subUint(bytes32,uint256)._amount (RocketTokenRETH.sol#839) is not in mixedCase
Constant RocketBase.calcBase (RocketTokenRETH.sol#691) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter RocketTokenRETH.getEthValue(uint256)._rethAmount (RocketTokenRETH.sol#894) is not in mixedCase
Parameter RocketTokenRETH.getRethValue(uint256)._ethAmount (RocketTokenRETH.sol#906) is not in mixedCase
Parameter RocketTokenRETH.mint(uint256,address)._ethAmount (RocketTokenRETH.sol#949) is not in mixedCase
Parameter RocketTokenRETH.mint(uint256,address)._to (RocketTokenRETH.sol#949) is not in mixedCase
Parameter RocketTokenRETH.burn(uint256)._rethAmount (RocketTokenRETH.sol#961) is not in mixedCase
Parameter RocketTokenRETH.withdrawDepositCollateral(uint256)._ethRequired (RocketTokenRETH.sol#981) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (RocketTokenRETH.sol#404)" inContext (RocketTokenRETH.sol#398-407)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in RocketTokenRETH.burn(uint256) (RocketTokenRETH.sol#961-978):
        External calls:
        - msg.sender.transfer(ethAmount) (RocketTokenRETH.sol#975)
        Event emitted after the call(s):
        - TokensBurned(msg.sender,_rethAmount,ethAmount,block.timestamp) (RocketTokenRETH.sol#977)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

RocketBase.rocketStorage (RocketTokenRETH.sol#697) should be immutable
RocketBase.version (RocketTokenRETH.sol#694) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
RocketTokenRETH.sol analyzed (11 contracts with 84 detectors), 85 result(s) found
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Solidity Static Analysis

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

RocketTokenRETH.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in RocketTokenRETH.burn(uint256): Could potentially lead to re-entrancy vulnerability.
Note: Modifiers are currently not considered by this static analysis.
more
Pos: 961:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 793:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 957:55:

Gas costs:

Gas requirement of function RocketTokenRETH.burn is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 961:4:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function RocketTokenRETH.depositExcessCollateral is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 991:4:

Constant/View/Pure functions:

RocketTokenRETH.getEthValue(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 894:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 650:8:

No return:

RocketNetworkBalancesInterface.getTotalETHBalance(): Defines a return type but never explicitly returns a value.
Pos: 380:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 266:15:

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

RocketTokenRETH.sol

```
Compiler version >=0.6.0 <0.8.0 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:2
Error message for require is too long
Pos: 9:192
Error message for require is too long
Pos: 9:584
Error message for require is too long
Pos: 9:585
Error message for require is too long
Pos: 9:625
Error message for require is too long
Pos: 9:648
Error message for require is too long
Pos: 9:649
Code contains empty blocks
Pos: 94:680
Explicitly mark visibility of state
Pos: 5:690
Constant name must be in capitalized SNAKE_CASE
Pos: 5:690
Explicitly mark visibility of state
Pos: 5:696
Error message for require is too long
Pos: 9:705
Error message for require is too long
Pos: 9:746
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:756
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:796
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:881
Avoid making time-based decisions in your business logic
Pos: 52:889
Error message for require is too long
Pos: 9:913
Avoid making time-based decisions in your business logic
Pos: 52:943
Avoid making time-based decisions in your business logic
Pos: 56:956
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

```
Error message for require is too long
Pos: 9:968
Avoid making time-based decisions in your business logic
Pos: 63:976
Error message for require is too long
Pos: 17:1021
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.
Email: audit@EtherAuthority.io

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io