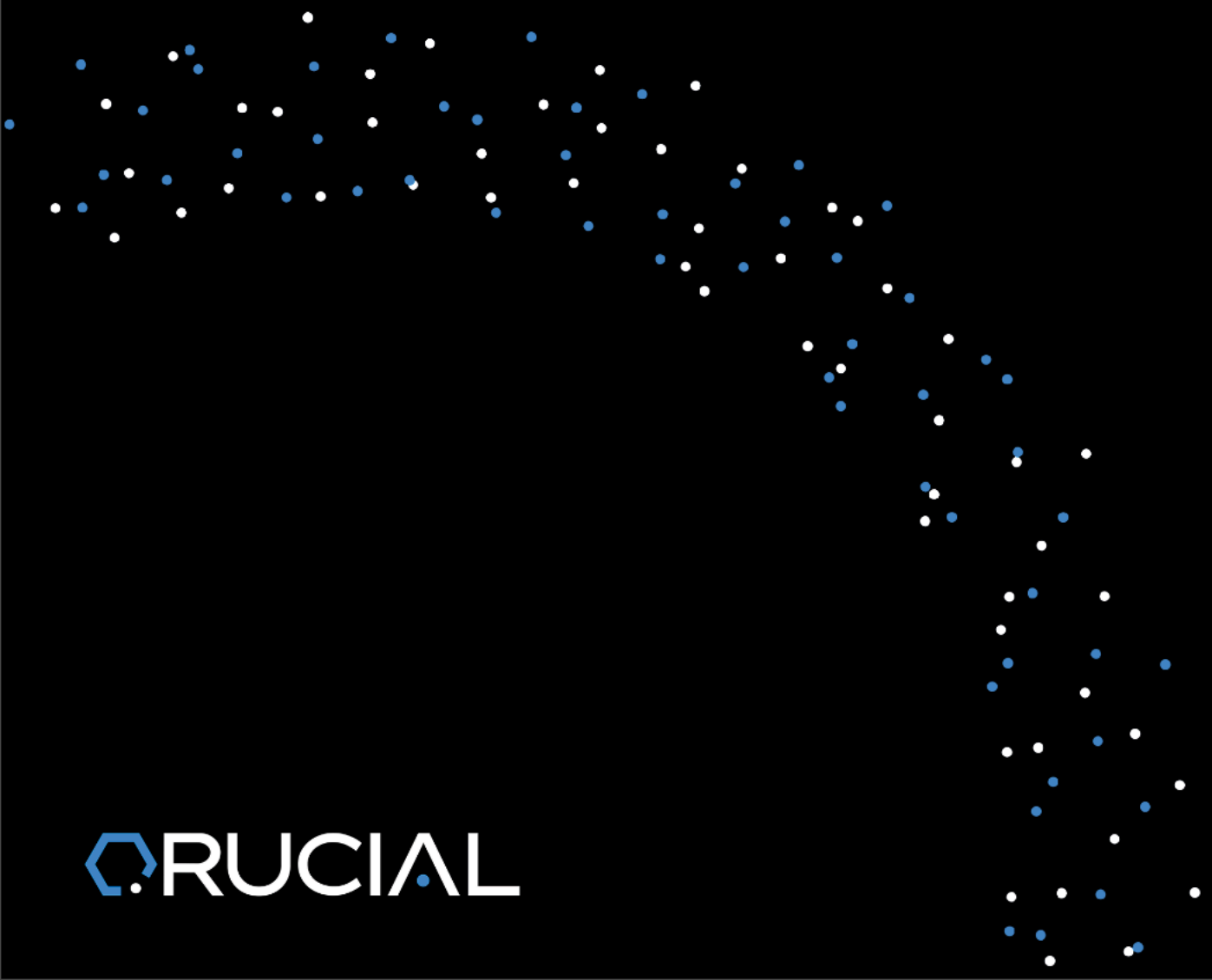
SMART CONTRACT SECURITY REPORT.

Personally audited and created for:
LGBTQ Coin.



CRUCIAL

21th of December 2021

TABLE OF CONTENTS

Introduction

Overview

- Project
- Audit
- Vulnerability
- Audit Scope

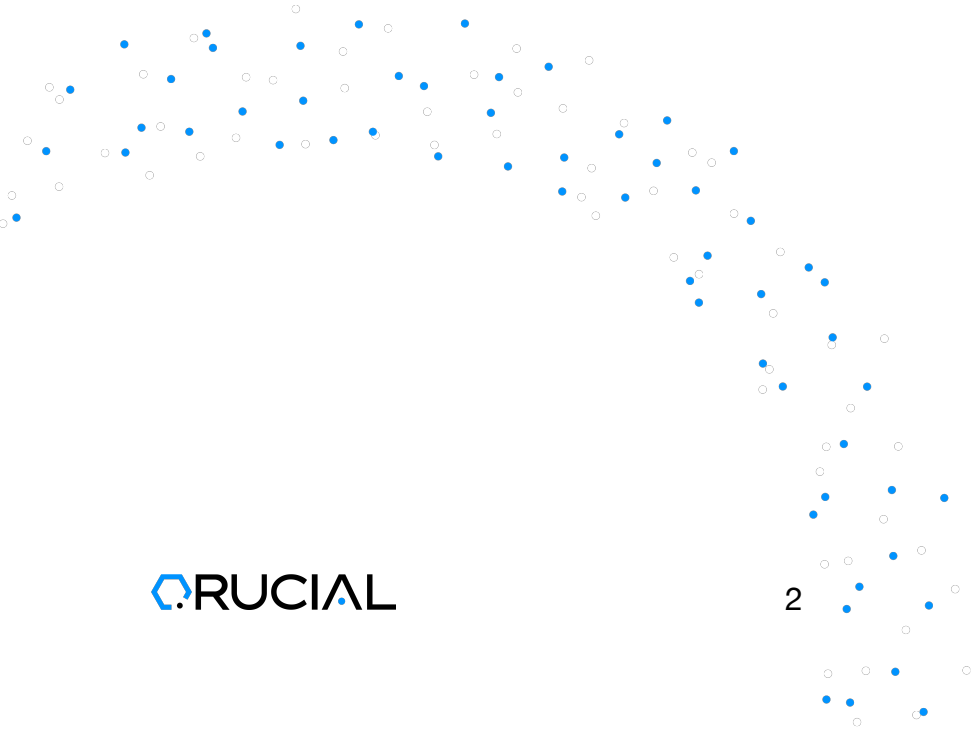
Risk classifications

Findings

Appendix

Disclaimer

Thank you



Introduction

This report has been prepared for LGBTQ Coin. An extensive analysis has been performed by manual review, static analysis and symbolic execution.

In our Audits, we focus on the following areas:

- Analyzing that the smart contract's logic meet the intentions of the client.
- Examining the smart contracts against conventional and unconventional attack vectors
- Verifying the codebase meets the current Industry standard and best practices
- Cross-referencing the Project against the implementation, contract, and structure of similar Industry-leading Projects
- Elaborate line by line manual review of the entire Codebase.

The findings of the security evaluation resulted ranged from medium to informational.
We advise you to address these findings as soon as possible to assure a foremost level of security for your project and community.

- Increase good coding practices for a better structure in the source code
- Increase the number of unit test to cover all possible angles of use cases
- Add more comments per function for readability.

Overview

Project Summary

Project name	LGBTQ Coin
Platform	Polygon
Language	Solidity
Codebase	
SHA256 Sum	lgbtq.sol - 716f8305cfc83c1d71c26a4426ece628491387a825d1eb718178830ce962ea11

Audit Summary

International Crypto Trade Ltd. Contacted QRUCIAL OÜ in order to conduct a smart contract security audit for their LBGTQ Token.

The goal of this audit was to uncover potential security vulnerabilities to inspect its general architecture and design of the solidity implementation of the business model, as well as finding bugs which could imperil the software in production. The findings of the initial Audit were sent to the Project's team and the source code is expected to be re-assessed before the live deployment.

QRUCIAL has identified **medium** and **informational** level findings.

Vulnerability Summary

Level	Total	Pending	Rejected	Accepted	Partially fixed	Fixed
Critical	0	-	-	-	-	-
High	0	-	-	-	-	-
Medium	1	-	-	1	-	-
Low	1	-	-	-	1	-
Informational	1	-	-	-	1	-

Scope

Audited Code:	LGBTQ Coin
Blockchain Explorer Link:	
Compiler:	0.8.9
Number of files:	1
Scope (list of files)	LGBTQ.sol

Risk Classifications

Critical:
Vulnerabilities that can lead to a loss of funds, impairment, or control over the system or its function.
We recommend that findings of this classification are fixed immediately.

High:
Findings of this classification can impact the flow of logic and can cause direct disruption in the system and the project's organization.
We recommend that issues of this classification are fixed as soon as possible.

Medium:
Vulnerabilities of this class have impact on the flow of logic, but does not cause any disturbance that would halt the system or organizational continuity.
We recommend that findings of this class are fixed nonetheless.

Low:
Bugs, or vulnerability that have minimal impact and do not pose a significant threat to the project or its users.
We recommend that issues of this class are fixed nonetheless because they increase the attack surface when your project is targeted by malicious actors.

Informational:
Findings of this class have a negligible risk factor but refer to best practices in syntax, style or general security.

MEDIUM: Lack of decentralization by a single point of failure

Description:
During the audit, it was found that the project has a single point of failure in the system: the smart contract can be upgraded and changed by a single account.

We consider this issue medium level because a lack of dezentralization opens attack vectors.

Impact:
In case the account is breached, the project might be taken down as a whole. It can happen through multiple scenarios, for example:

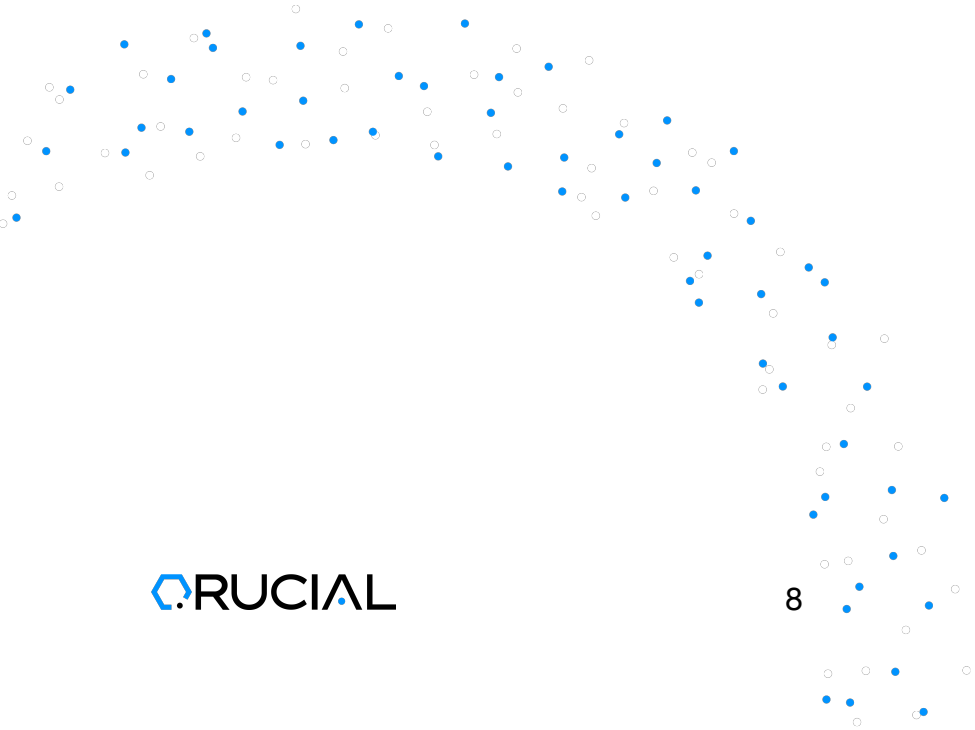
- Physical tampering or theft of the device that stores the private keys
- By human error, losing the device that stores private keys
- System error, eg. ssd/disk failure and lack of usable backup
- Insider threat
- Incident of the device owner and having no possibility to restore the private keys
- Natural disaster
- Phishing

Recommendations:
Implement a logic that requires multiple signatures to take actions like pausing the whole contract. An example can be threshold ESCDA.

References:
https://github.com/Qrucial/Voronoi
Threshold ECDSA: https://eprint.iacr.org/2019/114.pdf

Technical Details:

```
function transferWithVesting(address recipient, uint256 amount)
external onlyOwner
function setCharityAddr(address payable addr) external onlyOwner
function setMarketingAddr(address payable addr) external onlyOwner
function setTeamAddr(address payable addr) external onlyOwner
function jumpShip() external onlyOwner
```



LOW: Optimization - Visibility of functions could be stricter

Description:
If a function does not require to be called internally, it is possible to save gas costs and improve security by changing their visibility from public to external.

Note also that the previous high vulnerability might be easier to exploit if the visibility of these functions are public and are callable internally.

Impact:
Function calls will cost less gas (both deploy and call times) and security will be slightly improved.

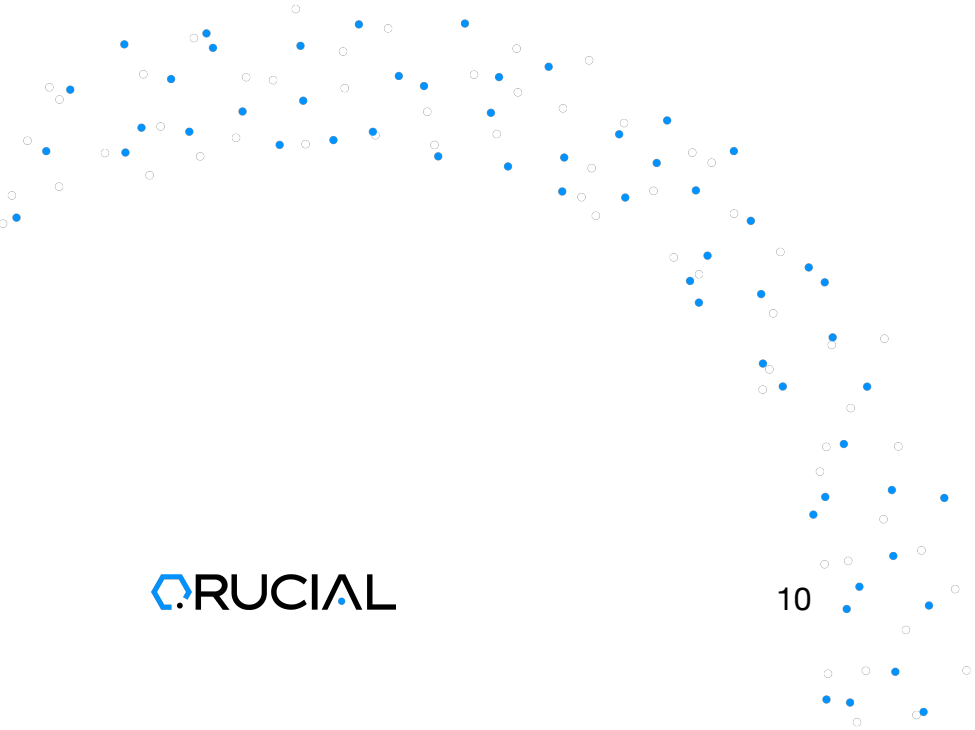
Recommendations:
Replace “public“ to “external“ in all functions listed above.

References:
https://ezcook.de/2018/01/29/Gas-Used-by-Public-and-External-Function-in-Solidity/

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Technical Details:

- Ownable.renounceOwnership() (contracts/access/Ownable.sol#54-56)
- Ownable.transferOwnership(address) (contracts/access/Ownable.sol#62-65)
- ERC777.name() (contracts/token/ERC777/ERC777.sol#81-83)
- ERC777.symbol() (contracts/token/ERC777/ERC777.sol#88-90)
- ERC777.decimals() (contracts/token/ERC777/ERC777.sol#98-100)
- ERC777.granularity() (contracts/token/ERC777/ERC777.sol#107-109)
- ERC777.send(address,uint256,bytes) (contracts/token/ERC777/ERC777.sol#130-136)
- ERC777.burn(uint256,bytes) (contracts/token/ERC777/ERC777.sol#165-167)
- ERC777.authorizeOperator(address) (contracts/token/ERC777/ERC777.sol#182-192)
- ERC777.revokeOperator(address) (contracts/token/ERC777/ERC777.sol#197-207)
- ERC777.defaultOperators() (contracts/token/ERC777/ERC777.sol#212-214)
- ERC777.operatorSend(address,address,uint256,bytes,bytes) (contracts/token/ERC777/ERC777.sol#221-230)
- ERC777.operatorBurn(address,uint256,bytes,bytes) (contracts/token/ERC777/ERC777.sol#237-245)
- ERC777.allowance(address,address) (contracts/token/ERC777/ERC777.sol#254-256)
- ERC777.approve(address,uint256) (contracts/token/ERC777/ERC777.sol#263-267)



Appendix

INFORMATIONAL: Assembly and low level calls

Description:
Using low-level calls and assembly in smart contracts highly increase project complexity and possibilities for error, hence they are be er to be avoided if not required.

Impact:
Even if not directly a vulnerability, low-level calls might open up the surface for high-impact attacks.

Recommendations:
Avoid low-level calls whenever possible.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

https://docs.soliditylang.org/en/v0.8.9/control-structures.html#error-handling-assert-require-revert-and-exceptions

Technical Details:

Low level call in LGBTQ.jumpShip() (lgbtq_087.sol#74-78):

- (success,None) = address(msg.sender).call{value: address(this).balance}()

(lgbtq_087.sol#77)

Low level call in Address.sendValue(address,uint256) (contracts/utils/Address.sol#60-65):

- (success) = recipient.call{value: amount}() (contracts/utils/Address.sol#63)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/utils/Address.sol#128-139):

- (success,returndata) = target.call{value: value}(data) (contracts/utils/Address.sol#137)

Low level call in Address.functionStaticCall(address,bytes,string) (contracts/utils/Address.sol#157-166):

- (success,returndata) = target.staticcall(data) (contracts/utils/Address.sol#164)

Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/utils/Address.sol#184-193):

- (success,returndata) = target.delegatecall(data) (contracts/utils/Address.sol#191)

DISCLAIMER

This report is fixed to the scope and subject to terms and conditions of the service agreement provided to the customer. This report must not be referred, transmitted or disclosed to a third party without QRUCIAL's prior written consent.

This report is not an endorsement disapproval of a team, a product, a service, a company, or an individual. This report should not be considered as financial advice and does not indicate any financial or economic value in an asset, an asset class a product or service. This report is not to be seen as an indication of the legal compliance regarding of a project an asset, an asset class or a business model.

This report does not provide the guarantee, that a project is without bugs, errors vulnerabilities or code that is harmful to machines, software, or data. This report is also no indication of the validity of any business model or technology. Each individual organization is responsible to do their own due diligence or security assessment. This report is not to be seen as a guarantee of the functionality of a technology or its security. The use of access or information in this audit is used on the risk of the reader or user of this document.

This report holds no guarantee that the given information meets requirements of any kind, is compatible with applications, any software or systems. It is also not guaranteed that this audit is free of errors or harmful code or will cause interruptions of any software or systems. We do not give any guarantee of accuracy, reliability, or correctness of the information given in this audit. All third-party material provided to the client may be subject to the terms and conditions of third parties. All third-party material provided is provided without the guarantee of correctness. No third-party has the right to use the trademark QRUCIAL, its products or services as a reference or endorsement of its own products or services without prior written consent.

THANK YOU.

Our team gave their full commitment to craft this audit with precision and focus on details with the goal to support you improving your work.

With this audit we want to provide you a guidance to make your project more
secure and for presenting your
community a product they can trust in.

We make security our priority, so you don‘t have to.

CRUCIAL

----Polkadot{.js} account validation address----

5EHagRYLNsCJUx5bA5Y8MZWLqPVzqQbFMC76V68Wzv7GHHXD

----Polkadot{.js} account validation address----

QRUCIAL

QRUCIAL OÜ c/o Gate to Baltics
Narva mnt 7-652, 10117 | Tallinn, Estonia