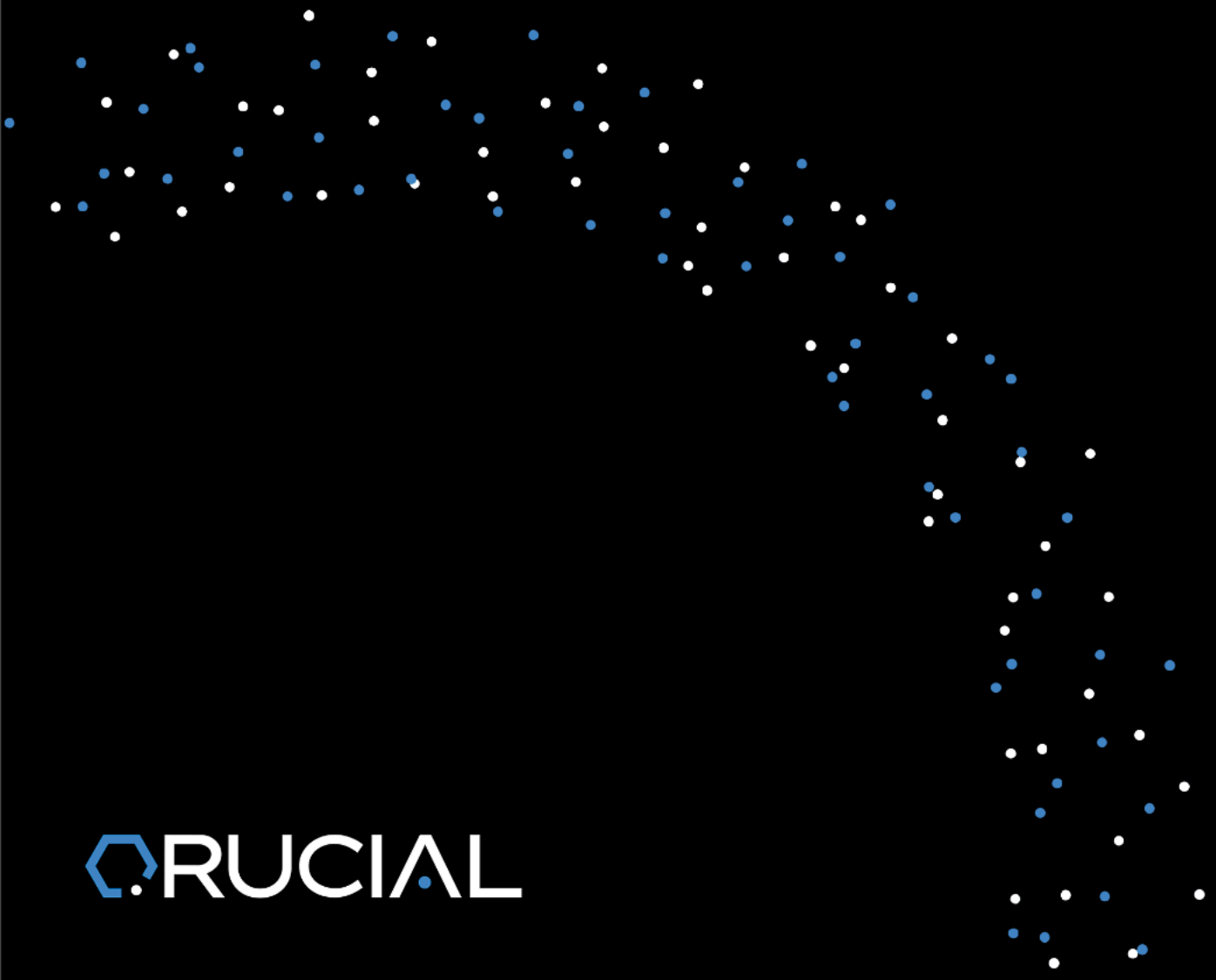
SMART CONTRACT SECURITY REPORT.

Personally audited and created for:
Wrapped Banano.



CRUCIAL

11th of November 2021

TABLE OF CONTENTS

Introduction

Overview

- Project
- Audit
- Vulnerability
- Audit Scope

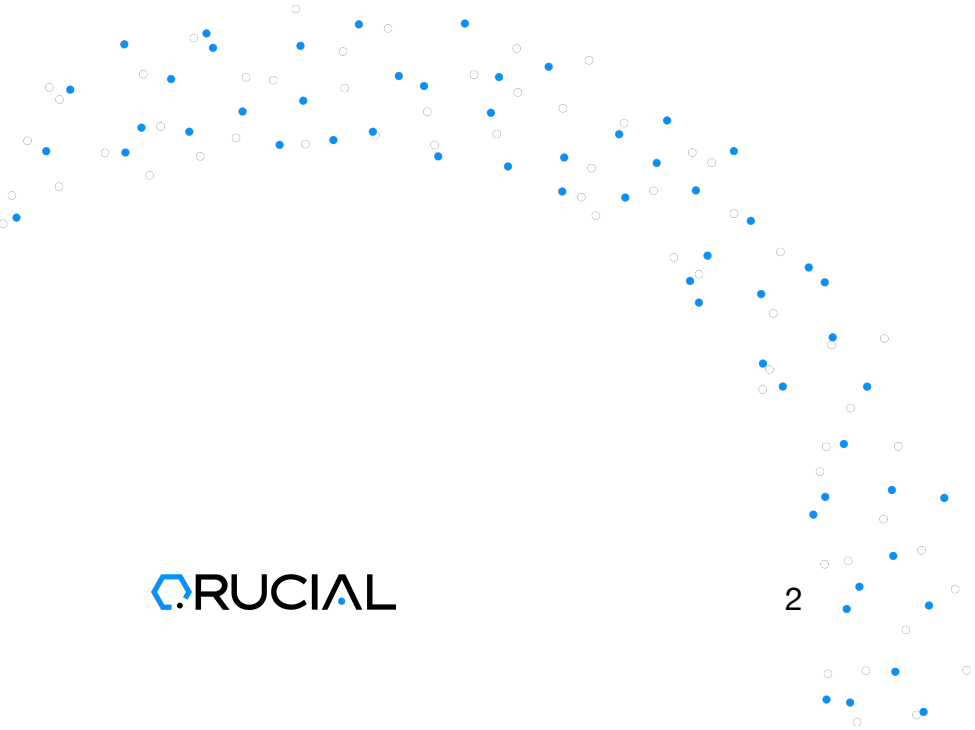
Risk classifications

Findings

Appendix

Disclaimer

Thank you



Introduction

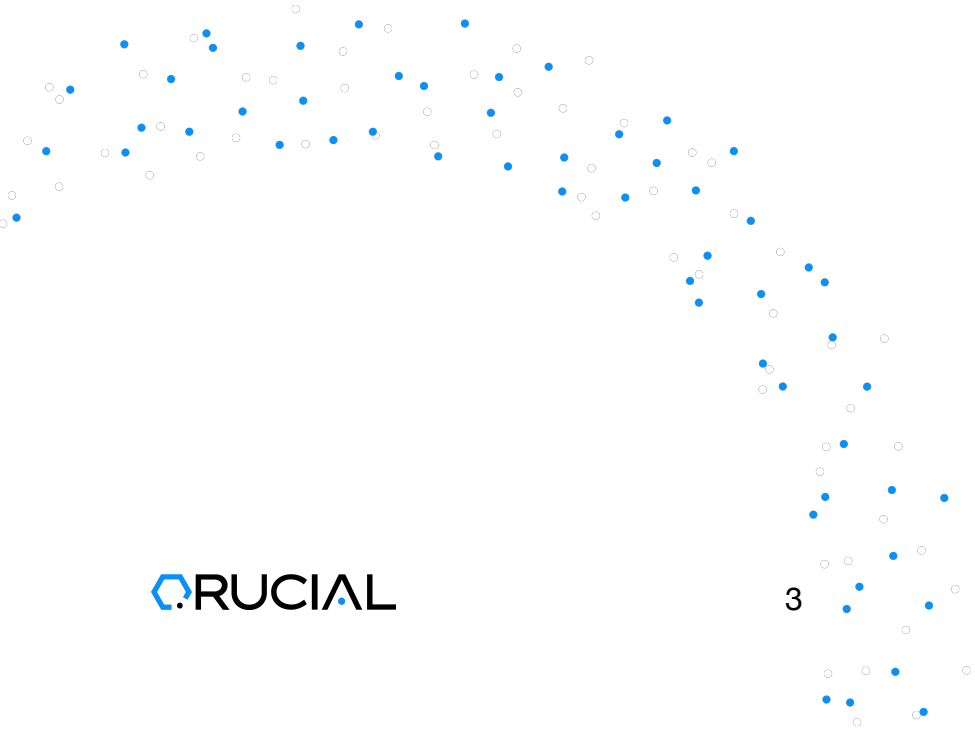
This report has been prepared for Wrapped Banano. An extensive analysis has been performed by manual review, static analysis and symbolic execution.

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	Wrapped Banano (wBAN)
Platform	Binance Chain
Language	Solidity
Codebase	https://bscscan.com/address/0xe20b9e246db5a0d21bf9209e4858bc9a3ff7a034#code
Commit	78f03722ec71679f93cea911628e888174c25cd0

Audit Summary

QRUCIAL has conducted a security audit on the Wrapped Banano project, after the wBan Bridge Rekt event (link:https://medium.com/banano/wrapped-banano-wban-bridges-rekt-post-mortem-80a66e1802d5). During this time, further vulnerabilities have been identified that can have an impact on project stability and health.

Wrapped Banano's principles include decentralization in the core, but the implementation does not follow it in multiple parts of the system. We recommend going through the findings and implement fixes.

Vulnerability Summary

Level	Total	Pending	Rejected	Accepted	Partially fixed	Fixed
Critical	0	-	-	-	-	-
High	0	-	-	-	-	-
Medium	1	-	-	-	-	-
Low	4	-	-	-	-	-
Informational	2	-	-	-	-	-

Scope

Audited Code:	Wrapped Banano
Blockchain Explorer Link:	https://bscscan.com/address/0xe20b9e246db5a0d21bf9209e4858bc9a3ff7a034#code
Compiler:	Compiler for the proxy contracts: v0.6.12 Compiler for the BEP20 contracts: v0.8.0
Number of files:	5
Scope (list of files)	AdminUpgradeabilityProxy.sol UpgradeabilityProxy.sol Proxy.sol Address.sol WBANToken.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.

– BEP20 implementation findings –

MEDIUM: Lack of decentralization by a single point of failure

Description:
During the audit it was found that the Wrapped Banano project has a single point of failure in the system.

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

- Stealing the device physically that stores the private keys
- Exploitation of the system that stores the private keys
- By human error, losing the device that stores private keys
- System error, e.g. ssd/disk failure and lack of useable backup
- Insider threat
- Incident of the device owner and having no possibility to restore the private keys

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

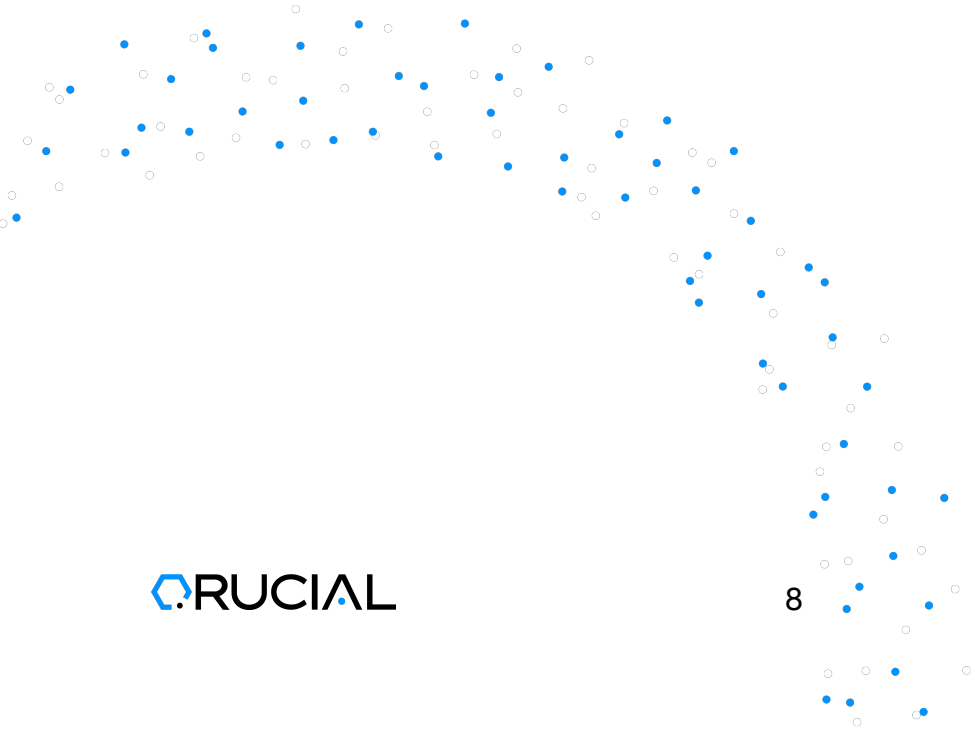
References:
https://github.com/Qrucial/Voronoi

Threshold ECDSA: https://eprint.iacr.org/2019/114.pdf

Technical Details:

The following roles are centralized to a single account address.

0x65d7a28e3265b37a6474929f336521b332c1681b933f6cb9f3376673440d862a - PAUSER_ROLE
0x9f2df0fed2c77648de5860a4cc508cd0818c85b8b8a1ab4ceeef8d981c8956a6 - MINTER
0x0000000000000000000000000000000000000000000000000000000000000000 - DEFAULT_ADMIN_ROLE



LOW: State Variable Shadowing

Description:
State variable names are shadowing themselves. We could not find a ways to exploit it in a meaningful way, but it is better to remove the shadowing to improve code quality.

Impact:
Increased complexity, making development more error prone.

Recommendations:
Rename variables and remove the shadowing. This makes also the logic more clear.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

Technical Details:

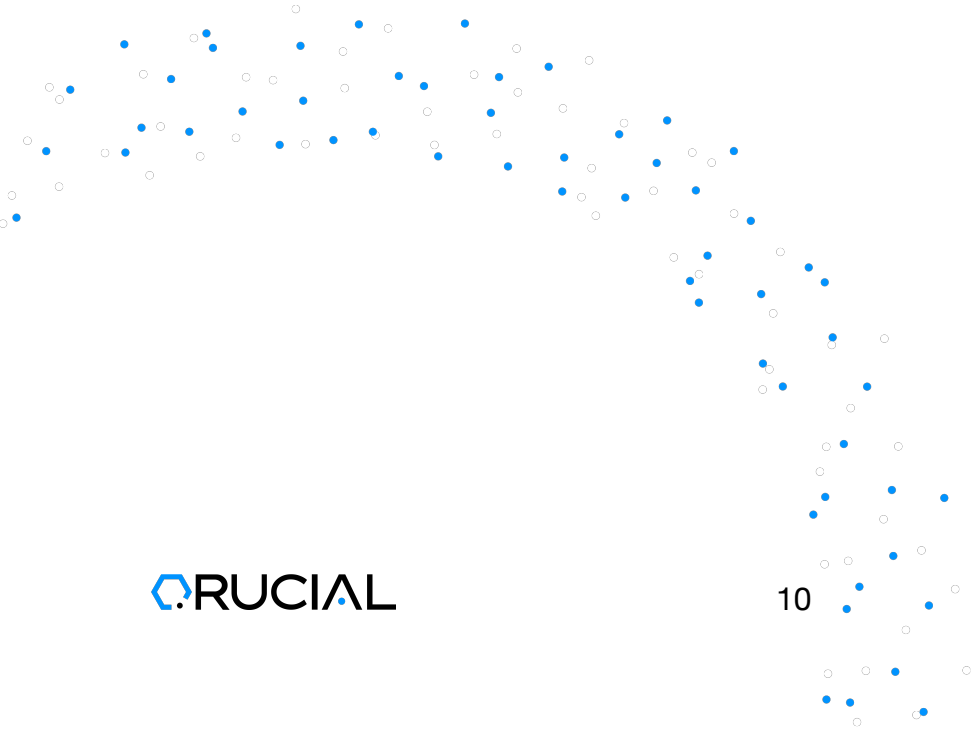
```
AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#259) shadows:
- ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35)
- ContextUpgradeable.__gap (ContextUpgradeable.sol#30)

ERC20PausableUpgradeable.__gap (ERC20PausableUpgradeable.sol#41) shadows:
- PausableUpgradeable.__gap (PausableUpgradeable.sol#96)
- ERC20Upgradeable.__gap (ERC20Upgradeable.sol#360)
- ContextUpgradeable.__gap (ContextUpgradeable.sol#30)

ERC20Upgradeable.__gap (ERC20Upgradeable.sol#360) shadows:
- ContextUpgradeable.__gap (ContextUpgradeable.sol#30)

OwnableUpgradeable.__gap (OwnableUpgradeable.sol#77) shadows:
- ContextUpgradeable.__gap (ContextUpgradeable.sol#30)

PausableUpgradeable.__gap (PausableUpgradeable.sol#96) shadows:
- ContextUpgradeable.__gap (ContextUpgradeable.sol#30)
```



LOW: Unused code and state variables

Description:
The more functions are available, the larger the attack surface is. Also, the code is more complex by that, hence the chance for bugs, errors are increased.

Impact:
Increased complexity and attack surface.

Recommendations:
Remove the unused code by upgrading the contract. We recommend making a contract upgrade that fixes all vulnerabilities together.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Technical Details:

```
OwnableUpgradeable.__gap (OwnableUpgradeable.sol#77) is never used in WBANToken
(WBANToken.sol#15-122)

List of related state variables
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32)
(AccessControlUpgradeable.sol#241-244)
ContextUpgradeable.__Context_init() (ContextUpgradeable.sol#17-19)
ContextUpgradeable._msgData() (ContextUpgradeable.sol#27-29)

ERC165Upgradeable.__ERC165_init() (ERC165Upgradeable.sol#23-25)

ERC20PausableUpgradeable.__ERC20Pausable_init()
(ERC20PausableUpgradeable.sol#17-21)

ERC20PausableUpgradeable.__ERC20Pausable_init_unchained()
(ERC20PausableUpgradeable.sol#23-24)

ERC20PausableUpgradeable._beforeTokenTransfer(address,address,uint256)
(ERC20PausableUpgradeable.sol#32-40)

PausableUpgradeable.__Pausable_init() (PausableUpgradeable.sol#33-36)

PausableUpgradeable.__Pausable_init_unchained() (PausableUpgradeable.sol#38-40)

SafeMathUpgradeable.add(uint256,uint256) (SafeMathUpgradeable.sol#92-94)

SafeMathUpgradeable.div(uint256,uint256) (SafeMathUpgradeable.sol#134-136)

SafeMathUpgradeable.div(uint256,uint256,string)
(SafeMathUpgradeable.sol#190-199)

SafeMathUpgradeable.mod(uint256,uint256) (SafeMathUpgradeable.sol#150-152)

SafeMathUpgradeable.mod(uint256,uint256,string)
(SafeMathUpgradeable.sol#216-225)

SafeMathUpgradeable.mul(uint256,uint256) (SafeMathUpgradeable.sol#120-122)

SafeMathUpgradeable.sub(uint256,uint256) (SafeMathUpgradeable.sol#106-108)

SafeMathUpgradeable.sub(uint256,uint256,string)
(SafeMathUpgradeable.sol#167-176)

SafeMathUpgradeable.tryAdd(uint256,uint256) (SafeMathUpgradeable.sol#21-27)

SafeMathUpgradeable.tryDiv(uint256,uint256) (SafeMathUpgradeable.sol#63-68)

SafeMathUpgradeable.tryMod(uint256,uint256) (SafeMathUpgradeable.sol#75-80)

SafeMathUpgradeable.tryMul(uint256,uint256) (SafeMathUpgradeable.sol#46-56)

SafeMathUpgradeable.trySub(uint256,uint256) (SafeMathUpgradeable.sol#34-39)

StringsUpgradeable.toHexString(uint256) (StringsUpgradeable.sol#39-50)

StringsUpgradeable.toString(uint256) (StringsUpgradeable.sol#14-34)
```

– Proxy implementation findings –

LOW: Ether locking

Description:
Ether/Binance coin can be sent to the contract, but gets locked there.

Impact:
Coins can get locked in the contract, however it is unlikely anyone would actually send coins to this contract address.

Recommendations:
Stay aware of this logic when implementing the next contracts.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Technical Details:

```
Contract locking ether found:

Contract Proxy (Proxy.sol#12-77) has payable functions:
        - Proxy.fallback() (Proxy.sol#17-19)
        - Proxy.receive() (Proxy.sol#25-27)
       But does not have a function to withdraw the ether
```

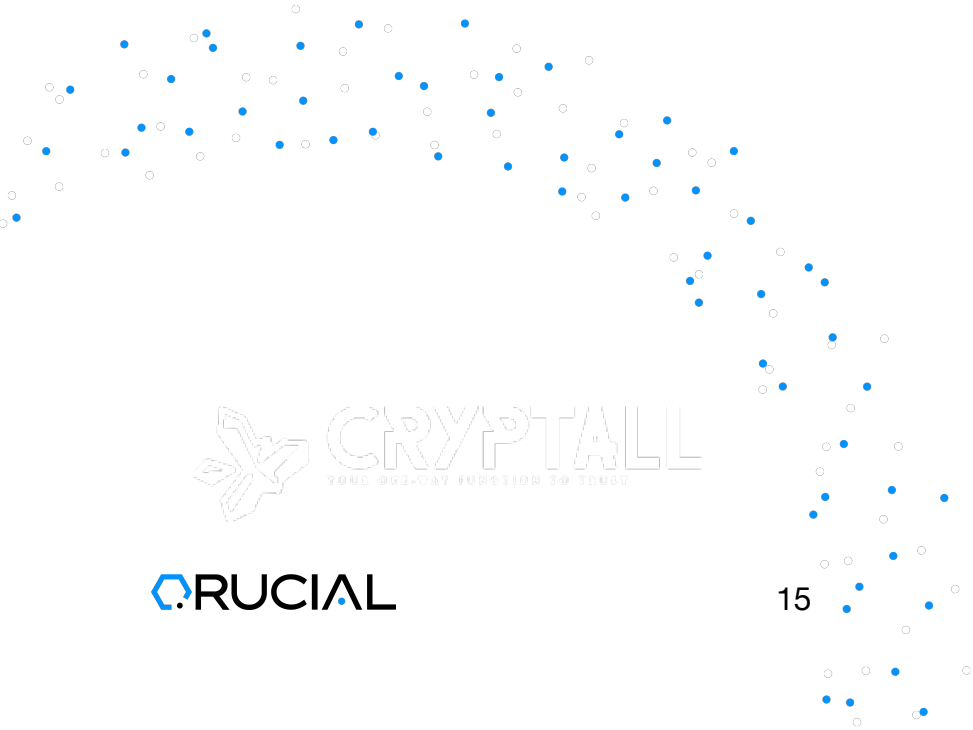
LOW: Unused code in the contract

Description:
The more functions are available, the larger the a ack surface is. Also, the code is more complex by that, hence the chance for bugs, errors are increased.

Impact:
Increased complexity and attack surface.

Recommendations:
Remove the unused code by upgrading the contract. We recommend making a contract upgrade that xes all vulnerabilities together.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code



Technical Details:

Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) is never used and should be removed

Address.functionCall(address,bytes) (Address.sol#79-81) is never used and should be removed

Address.functionCall(address,bytes,string) (Address.sol#89-91) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (Address.sol#104-106) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol#114-121) is never used and should be removed

Address.functionDelegateCall(address,bytes) (Address.sol#153-155) is never used and should be removed

Address.functionDelegateCall(address,bytes,string) (Address.sol#163-169) is never used and should be removed

Address.functionStaticCall(address,bytes) (Address.sol#129-131) is never used and should be removed

Address.functionStaticCall(address,bytes,string) (Address.sol#139-145) is never used and should be removed

Address.isContract(address) (Address.sol#26-35) is never used and should be removed

Address.sendValue(address,uint256) (Address.sol#53-59) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

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:

```
Address.isContract(address) (Address.sol#26-35) uses assembly
       - INLINE ASM (Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) uses assembly
        - INLINE ASM (Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#assembly-usage


Low level call in Address.sendValue(address,uint256) (Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data)
(Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string)
(Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string)
(Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-
level-calls


Proxy._delegate(address) (Proxy.sol#40-59) uses assembly
        - INLINE ASM (Proxy.sol#41-58)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#assembly-usage
```

INFORMATIONAL: Broad pragma version

Description:
The difference between solidity 0.6.2 and 0.8.0 of solidity is too high.

Impact:
The way the compiled byte codes behave are highly different before and at or after solidity 0.8.0.

Recommendations:
Specify more exact pragma.

References:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Technical Details:

```
Pragma version>=0.6.2<0.8.0 (Address.sol#3) is too complex
```

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.

RUCIAL

----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