SMART CONTRACT SECURITY REPORT.

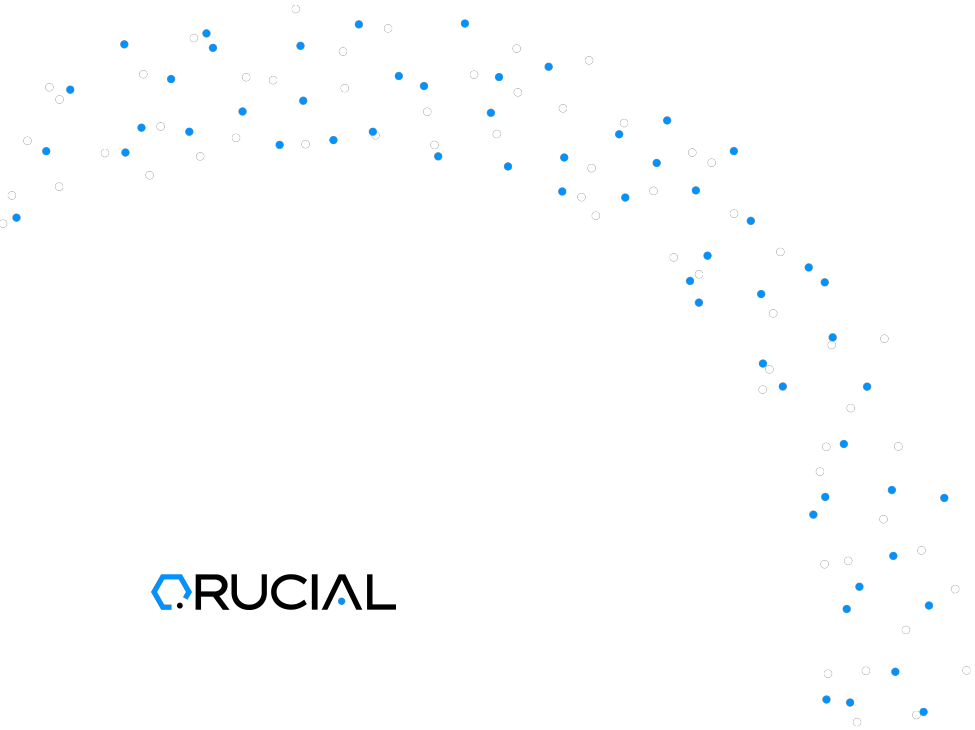
Personally audited and created for:
FPS Token.

CRUCIAL

CRYPTALL
YOUR ONE-WAY FUNCTION TO TRUST

17th of December 2021

TABLE OF CONTENTS



Introduction

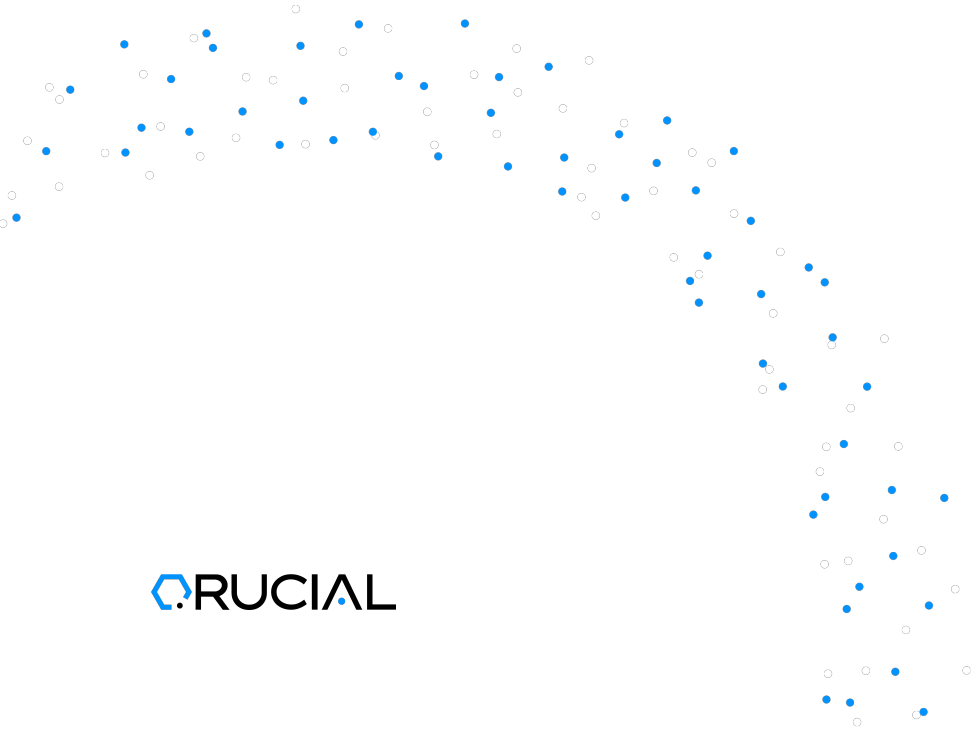
This report has been prepared for FPS Token. An extensive analysis has been performed by manual review, Static Analysis and symbolic execution.

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	FPS Token
Platform	Binance Chain
Language	Solidity
Codebase	https://bscscan.com/address/0x213503534d927424a5cdf6d580e9fd408be9337a
Commit	

Audit Summary

QRUCIAL's team was contacted by FPS Token to audit the logic, design as well as the implementation of the FPS Token smart contract. QRUCIAL security experts audited the FPS Token smart contracts in cooperation with cryptall against possible vulnerabilities and logic bugs that can lead to financial or reputational loss.

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 a second round of auditing is being carried out.

Vulnerability Summary

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

Scope

Audited Code:	FPS Token
Blockchain Explorer Link:	https://bscscan.com/address/0x213503534d927424a5cdf6d580e9fd408be9337a
Compiler:	^0.8.9
Number of files:	1
Scope (list of files)	FPS.sol

Risk Classifications

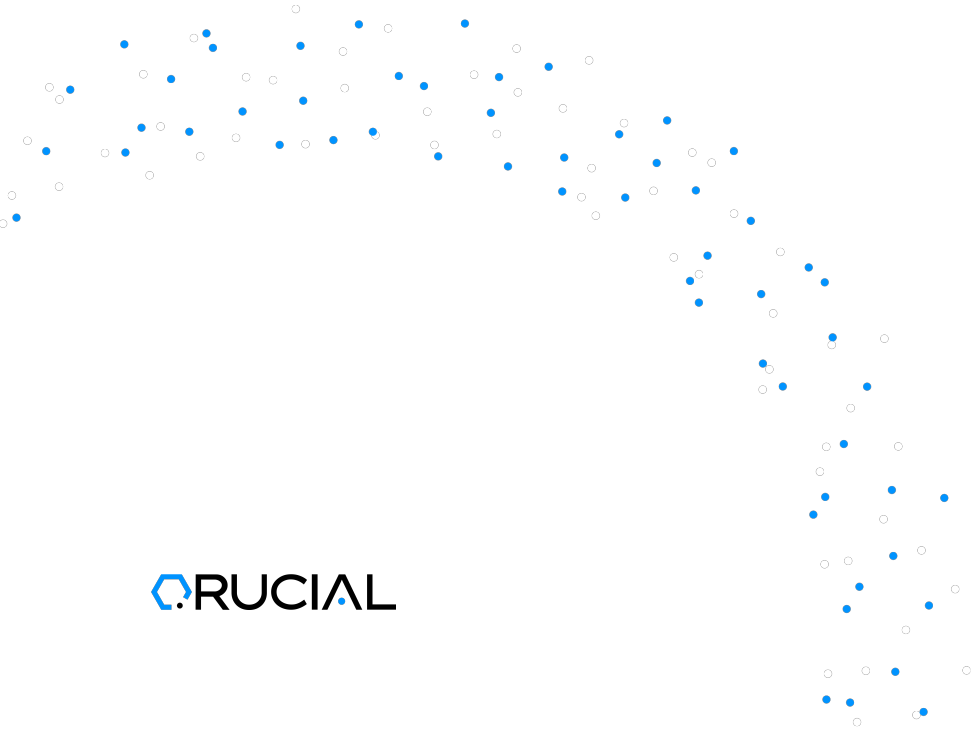
Critical:
Vulnerabilities that can lead to a loss of funds, impairment, or control over the system or its function.
We recommend that findings of this classification are fixed immediately.

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.

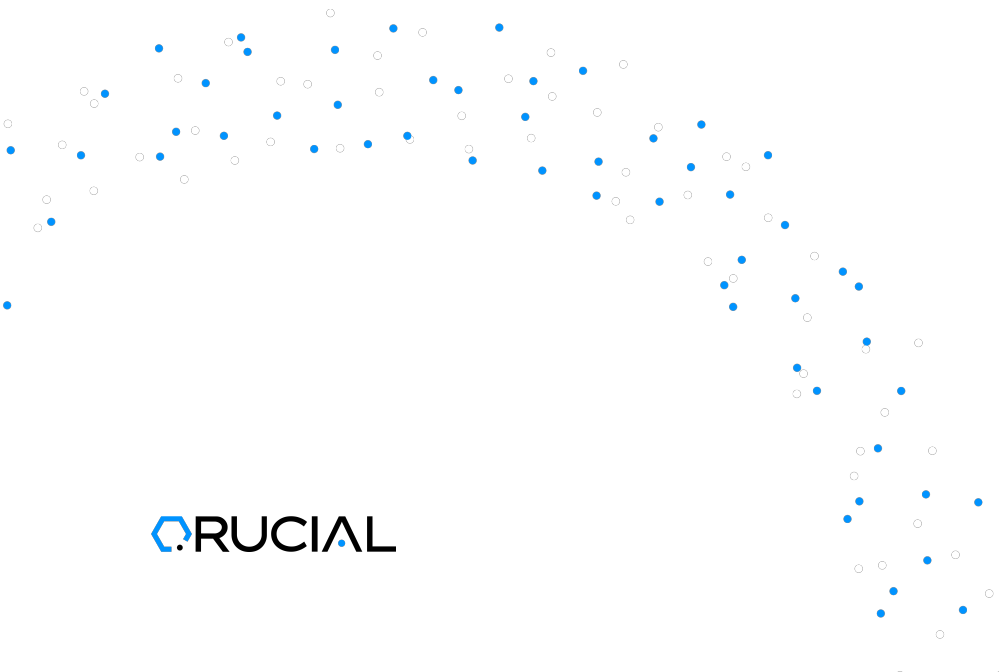
We consider this issue medium level because a lack of dezentralization opens attack vectors.

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

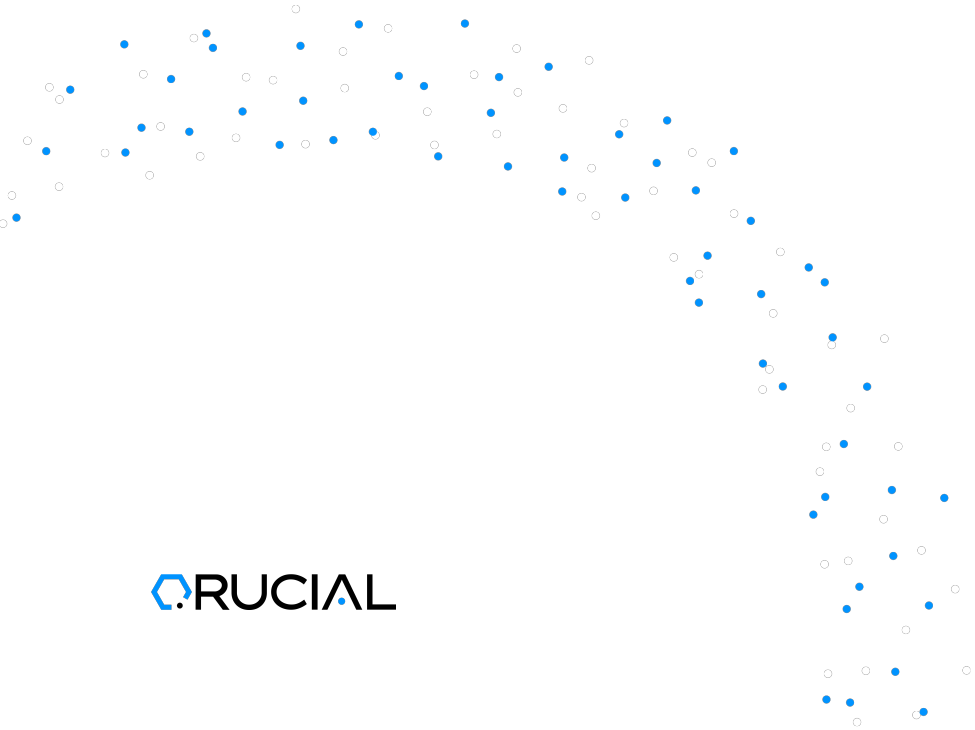
Recommendations:
Implement a logic that requires multiple signatures to take actions like pausing the whole contract. An example can be threshold ESCDA.

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



Technical Details:

```
The following roles are centralized to a single account address.
function renounceOwnership() / public / onlyOwner
function transferOwnership(address newOwner) / public / onlyOwner
function excludeFromMaxTransaction(address updAds, bool isEx) / public /
onlyOwner
function setAutomatedMarketMakerPair(address pair, bool value) public onlyOwner
function excludeFromReward(address account) / public / onlyOwner
function includeInReward(address account) / public / onlyOwner
function setLiquidityAddress(address _liquidityAddress) / public / onlyOwner
function setSwapAndLiquifyEnabled(bool _enabled) / public / onlyOwner
function removeLimits() / external / onlyOwner
function disableTransferDelay() / external / onlyOwner
function enableTrading() / external / onlyOwner
function updateMinimumTokensBeforeSwap(uint256 newAmount)  / external /
onlyOwner
function updateMaxAmount(uint256 newNum) / external / onlyOwner
function setGasPriceLimit(uint256 gas) / external / onlyOwner
function excludeFromFee(address account) / external / onlyOwner
function includeInFee(address account) / external / onlyOwner
function setMarketingAddress(address _marketingAddress) / external / onlyOwner
function setDevAddress(address _address) / external / onlyOwner
function setCommunityPrizeAddress(address _address) / external / onlyOwner
function withdrawStuckETH() / external / onlyOwner
```



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.

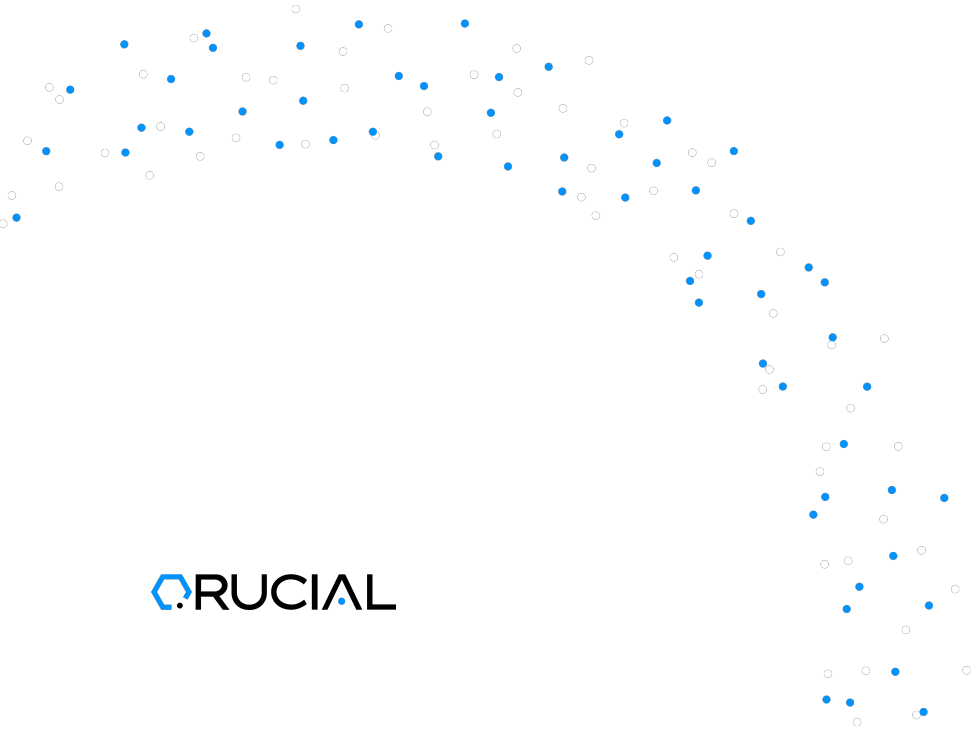
Impact:
Increased complexity and attack surface.

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:

```
Address._functionCallWithValue(address,bytes,uint256,string) (fps.sol#180-203)
Address.functionCall(address,bytes) (fps.sol#138-143)
Address.functionCall(address,bytes,string) (fps.sol#145-151)
Address.functionCallWithValue(address,bytes,uint256) (fps.sol#153-165)
Address.functionCallWithValue(address,bytes,uint256,string) (fps.sol#167-178)
Address.isContract(address) (fps.sol#111-122)
Address.sendValue(address,uint256) (fps.sol#124-136)
Context._msgData() (fps.sol#10-13)
SafeMath.mod(uint256,uint256) (fps.sol#96-98)
SafeMath.mod(uint256,uint256,string) (fps.sol#100-107)
```



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.

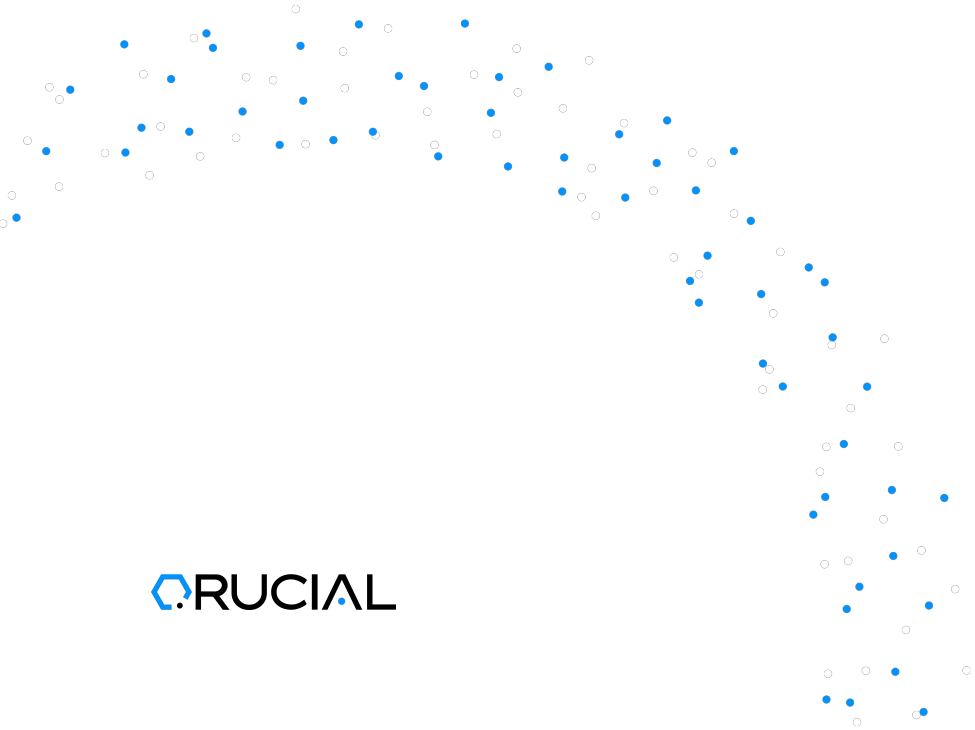
Recommendations:
Replace “public“ to “external“ in all functions listed above.

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() (fps.sol#231-234)
- Ownable.getUnlockTime() (fps.sol#245-247)
- Ownable.getTime() (fps.sol#249-251)
- FPSToken.approve(address,uint256) (fps.sol#570-577)
- FPSToken.setAutomatedMarketMakerPair(address,bool) (fps.sol#678-682)
- FPSToken.setLiquidityAddress(address) (fps.sol#1232-1237)
- FPSToken.setSwapAndLiquifyEnabled(bool) (fps.sol#1238-1241)



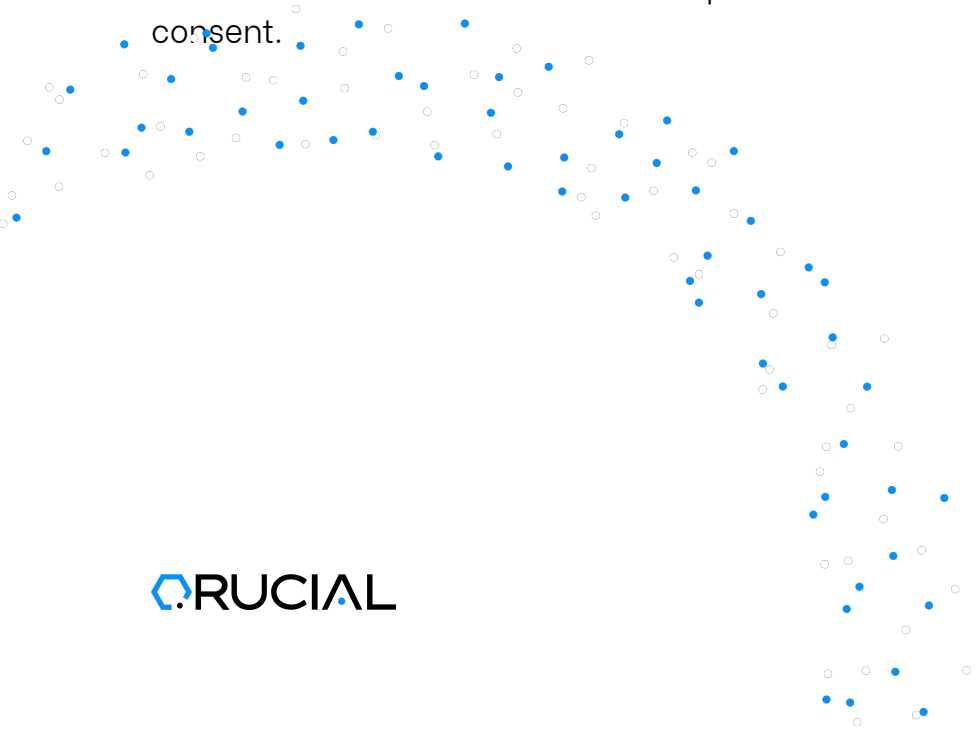
DISCLAIMER

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

CRYPTALL

YOUR ONE-WAY FUNCTION TO TRUST

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

5EHagRYLNsCJUx5bA5Y8MZWLqPVzqQbFMC76V68Wzv7GHHXD

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

CRYPTALL
YOUR ONE-WAY FUNCTION TO TRUST

QRUCIAL

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