SMART CONTRACT SECURITY REPORT.

Personally audited and created for:
TokoCrypto Token.

29th of November 2021

CRUCIAL

TABLE OF CONTENTS

Introduction

This report has been prepared for TokoCrypto 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	TokoCrypto Token
Platform	Binance Chain
Language	Solidity
Codebase	https://bscscan.com/address/0x9f589e3eabe42ebc94a44727b3f3531c0c877809#code
Commit	

Audit Summary

QRUCIAL OÜ was assessing the security of TokoCrypto Token (TKO) project. Based on the findings, no vulnerability has been found that could cause disruption in the TokoCrypto Token system, but there are several findings which can be considered in future smart contract development.

We advise to go through the findings and implement the recommendations in the upcoming projects.

Vulnerability Summary

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

Scope

Audited Code:	TokoCrypto Token
Blockchain Explorer Link:	https://bscscan.com/address/0x9f589e3eabe42ebc94a44727b3f3531c0c877809#code
Compiler:	v0.6.12
Number of files:	1
Scope (list of files)	contract.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: Visibility of functions should 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:

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (toko.sol#81-84)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (toko.sol#90-92)
decimals() should be declared external:
        - BEP20.decimals() (toko.sol#636-638)
symbol() should be declared external:
        - BEP20.symbol() (toko.sol#643-645)
totalSupply() should be declared external:
        - BEP20.totalSupply() (toko.sol#650-652)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (toko.sol#669-672)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (toko.sol#677-679)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (toko.sol#688-691)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (toko.sol#705-717)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (toko.sol#731-734)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (toko.sol#750-757)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (toko.sol#767-770)
mint(address,uint256) should be declared external:
        - TKOToken.mint(address,uint256) (toko.sol#885-888)
burn(address,uint256) should be declared external:
        - TKOToken.burn(address,uint256) (toko.sol#890-893)
```

LOW: Unused code in the contract

Description:
The more functions are available, the larger the attack surface is. Also the code is longer and more complex, hence it is harder to review.

Impact:
Increased complexity, deploy time gas fees and attack surface.

Recommendations:
Consider not deploying unused code in the future.

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

Technical Details:

```
Address._functionCallWithValue(address,bytes,uint256,string) (toko.sol#532-558)
Address.functionCall(address,bytes) (toko.sol#479-481)
Address.functionCall(address,bytes,string) (toko.sol#489-495)
Address.functionCallWithValue(address,bytes,uint256) (toko.sol#508-514)
Address.functionCallWithValue(address,bytes,uint256,string) (toko.sol#522-530)
Address.isContract(address) (toko.sol#424-435)
Address.sendValue(address,uint256) (toko.sol#453-459)
Context._msgData() (toko.sol#24-27)
SafeMath.div(uint256,uint256) (toko.sol#315-317)
SafeMath.div(uint256,uint256,string) (toko.sol#331-341)
SafeMath.min(uint256,uint256) (toko.sol#380-382)
SafeMath.mod(uint256,uint256) (toko.sol#355-357)
SafeMath.mod(uint256,uint256,string) (toko.sol#371-378)
SafeMath.mul(uint256,uint256) (toko.sol#289-301)
SafeMath.sqrt(uint256) (toko.sol#385-396)
```

Appendix

INFORMATIONAL: Dependence on on block.number for voting logic

Description:
Though not a vulnerability, it is better to be addressed as a potential point of danger as decision is made based on block.number environment variable.

Impact:
No exploitable impact known.

Recommendations:
This is an informational finding, there is no action needed..

References:
https://blog.sigmaprime.io/solidity-security.html#block-timestamp

Technical Details:

```
File: toko.sol
Line: 1033
    require(blockNumber < block.number, "TKO::getPriorVotes: not yet
determined")
```

```
File: toko.sol
Line: 1003
    require(now <= expiry, "TKO::delegateBySig: signature expired")
```

INFORMATIONAL: Overflows and underflows in the returns of functions

Description:
Though not a vulnerability, it is better to be addressed. The listed returns can overflow and underflow.

Impact:
The worst case scenario is misinformation returned to the user calling the function.

Recommendations:
This is an informational finding, there is no action needed.

References:
https://blog.sigmaprime.io/solidity-security.html#block-timestamp

Technical Details:

```
File: toko.sol
Line: 630
    return _name
```

```
File: toko.sol
Line: 644
    return _symbol
```

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

QRUCIAL

PROOFED

----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