VIRTUALCAIM

QUOTA

Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Assessment

May 2022

Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions, and recommendations set out in this publication are elaborated in the specific for only project.

Virtual Caim does not guarantee the authenticity of the project or organization or team of members that are connected/owner behind the project nor the accuracy of the data included in this study. All representations, warranties, undertakings, and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Company nor any person acting on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

Virtual Caim retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purposes of customers. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

Report Summary

Title	**QUOTA Smart Contract Audit**		
Project Owner	QUOTA		
Classification	Public		
Reviewed by	Virtual Caim Private Limited	Review date	13/05/2022
Approved by	Virtual Caim Private Limited	Approval date	13/05/2022
		Nº Pages	**25**

Overview

Background

QUOTA's team requested Virtual Caim to perform an Extensive Smart Contract Audit of their 'CoinToken' Smart Contract.

Project Dates

The following is the project schedule for this review and report:

- **May 13**: Smart Contract Review Started *(Completed)*
- **May 13**: Initial Delivery of Audit Findings *(Completed)*

Coverage

Target Specification and Revision

For this audit, we performed project's basic research, investigation by discussing the details with the project owner/developers, and then review the smart contract of QUOTA.

The following documentation & repositories were considered in -scope for the review:

QUOTA Project	https://bscscan.com/address/0xf223fb06766ad55272f179e59f1793ed8c27c706#code

Introduction

Given the opportunity to review QUOTA's Contract related smart contract source code, we in the report summary our methodical approach to evaluate all potential common security issues in the smart contract implementation, expose possible semantic irregularities between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to use after resolving the mentioned issues and done functional testing by owner/developer themselves, as there might be issues related to business logic, security or performance which only can found/understand by them.

About Audit

Item	Description
Issuer	QUOTA
Website	-
Type	ERC20
Platform	Binance
Language	Solidity
Audit Test Method	Whitebox Testing
Latest Audit Report	May 13, 2022

Test Methods Information

Test Methods

Black-box testing

Conduct security tests from an attacker's perspective externally.

Grey-box testing

Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.

White-box testing

Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

Vulnerability Severity Level Information

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

List of Check Items

Basic Coding Bugs

- Constructor Mismatch
- Ownership Takeover
- Redundant Fallback Function
- Overflows & Underflows
- Reentrancy
- MONEY-Giving Bug
- Blackhole
- Unauthorized Self-Destruct
- Revert DoS
- Unchecked External Call
- Gasless Send
- Send Instead of Transfer
- Costly Loop
- (Unsafe) Use of Untrusted Libraries
- (Unsafe) Use of Predictable Variables
- Transaction Ordering Dependence
- Deprecated Uses

Semantic Consistency Checks

- Semantic Consistency Checks

Advanced DeFi Scrutiny

- Business Logics Review
- Functionality Checks
- Authentication Management
- Access Control & Authorization
- Oracle Security
- Digital Asset Escrow
- Kill-Switch Mechanism
- Operation Trails & Event Generation
- ERC20 Idiosyncrasies Handling
- Frontend-Contract Integration
- Deployment Consistency
- Holistic Risk Management

Additional Recommendations

- Avoiding Use of Variadic Byte Array
- Using Fixed Compiler Version
- Making Visibility Level Explicit
- Making Type Inference Explicit
- Adhering To Function Declaration Strictly
- Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in this Audit

Category	Description
Configuration	• Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	• Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	• Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	• Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	• Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	• Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.

Smart Contract Audit

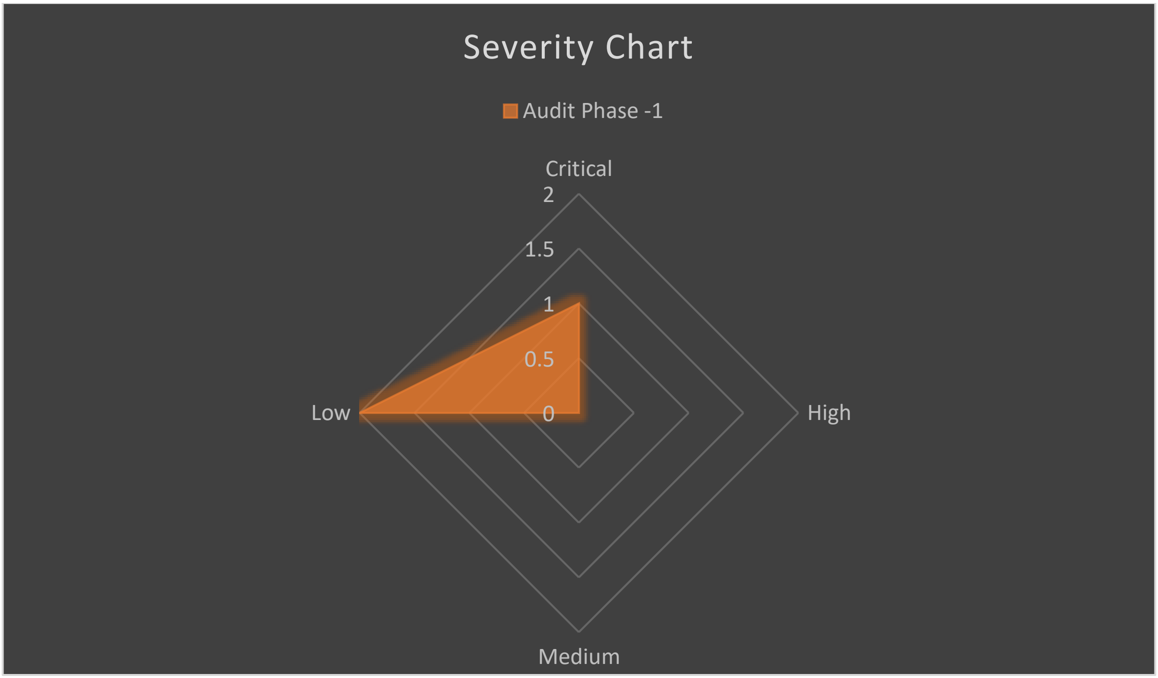
Resource Management	• Weaknesses in this category are related to improper management of system resources.
Behavioral Issues	• Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	• Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	• Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	• Weaknesses in this category are related to improper use arguments or parameters within function calls.
Expression Issues	• Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	• Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

Findings

Summary

Here is a summary of our findings after scrutinizing the QUOTA Smart Contract Review. During the first phase of our audit, we studied the smart contract source code and ran our in-house static code analyzer through the Specific tools. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tools. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	1
High	0
Medium	0
Low	2
Total	**3**



We have so far identified that there are potential issues with severity of **1 Critical, 0 High, 0 Medium, and 2 Low**. Overall, these smart contracts are well-designed and engineered.

Functional Overview

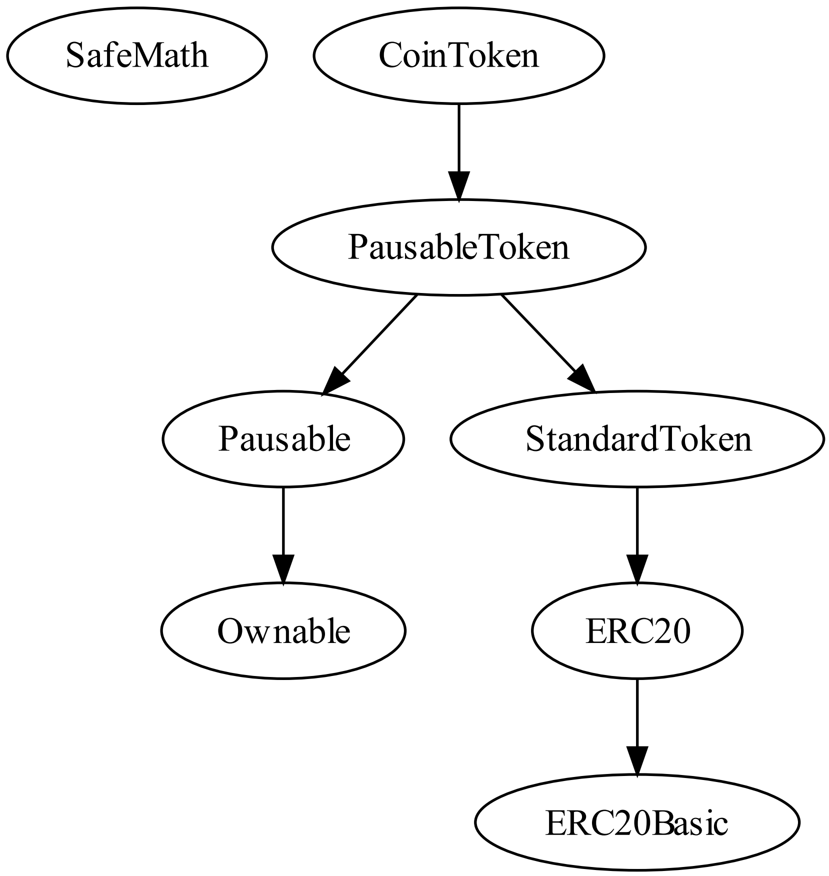
(\$) = payable function # = non-constant function	[Pub] public [Ext] external [Prv] private [Int] internal

```
 + [Lib] SafeMath
   - [Int] mul
   - [Int] div
   - [Int] sub
   - [Int] add

 +  Ownable
   - [Pub] transferOwnership #
      - modifiers: onlyOwner

 +  Pausable (Ownable)
   - [Pub] pause #
      - modifiers: onlyOwner,whenNotPaused
   - [Pub] unpause #
      - modifiers: onlyOwner,whenPaused
```

```
+ ERC20Basic
  - [Pub] balanceOf
  - [Pub] transfer #

+ ERC20 (ERC20Basic)
  - [Pub] allowance
  - [Pub] transferFrom #
  - [Pub] approve #

+ StandardToken (ERC20)
  - [Pub] transfer #
  - [Pub] balanceOf
  - [Pub] transferFrom #
  - [Pub] approve #
  - [Pub] allowance
  - [Pub] increaseApproval #
  - [Pub] decreaseApproval #
  - [Int] _blackList #

+ PausableToken (StandardToken, Pausable)
  - [Pub] transfer #
    - modifiers: whenNotPaused
  - [Pub] transferFrom #
    - modifiers: whenNotPaused
```

```
  - [Pub] approve #
    - modifiers: whenNotPaused
  - [Pub] increaseApproval #
    - modifiers: whenNotPaused
  - [Pub] decreaseApproval #
    - modifiers: whenNotPaused
  - [Pub] blackListAddress #
    - modifiers: whenNotPaused,onlyOwner

+  CoinToken (PausableToken)
  - [Pub] <Constructor> #
  - [Pub] burn #
  - [Int] _burn #
  - [Pub] mint #
    - modifiers: onlyOwner
```

Inheritance



Detailed Results

Issues Checking Status

1. Integer Overflow and Underflow

- SWC ID: 101
- Severity: Critical
- Location: CoinToken.sol
- Relationships: CWE-682: Incorrect Calculation
- Description: The arithmetic operator can overflow. It is possible to cause an integer overflow or underflow in the arithmetic operation.

```
    constructor(string memory _name, string memory _symbol, uint256 _decimals, uint256 _supply, address tokenOwner) public {
        name = _name;
        symbol = _symbol;
        decimals = _decimals;
        totalSupply = _supply * 10**_decimals;
        balances[tokenOwner] = totalSupply;
        owner = tokenOwner;
        emit Transfer(address(0), tokenOwner, totalSupply);
    }
```

- Remediations: It is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system.

2. **Floating Pragma**

- SWC ID: 103
- Severity: Low
- Location: CoinToken.sol
- Relationships: CWE-664: Improper Control of a Resource Through its Lifetime
- Description: A floating pragma is set. The current pragma Solidity directive is ""^0.4.24"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

```

pragma solidity ^0.4.24;

```

- Remediations: Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

3. State Variable Default Visibility

- SWC ID: 108
- Severity: Low
- Location: CoinToken.sol
- Relationships: CWE-710: Improper Adherence to Coding Standards
- Description: State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "tokenBlacklist", "balances" is internal. Other possible visibility settings are public and private.

```
mapping (address => mapping (address => uint256)) internal allowed;
      mapping(address => bool) tokenBlacklist;
      event Blacklist(address indexed blackListed, bool value);


mapping(address => uint256) balances;
```

- Remediations: Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

Automated Tool Results

Slither: -

```
CoinToken.constructor(string,string,uint256,uint256,address).tokenOwner (CoinToken.sol#238) lacks a zero-check on :
                - owner = tokenOwner (CoinToken.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

StandardToken.transfer(address,uint256) (CoinToken.sol#129-139) compares to a boolean constant:
        -require(bool)(tokenBlacklist[msg.sender] == false) (CoinToken.sol#130)
StandardToken.transferFrom(address,address,uint256) (CoinToken.sol#146-157) compares to a boolean constant:
        -require(bool)(tokenBlacklist[msg.sender] == false) (CoinToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

SafeMath.div(uint256,uint256) (CoinToken.sol#17-22) is never used and should be removed
SafeMath.mul(uint256,uint256) (CoinToken.sol#8-15) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.4.24 (CoinToken.sol#5) allows old versions
solc-0.4.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter StandardToken.transfer(address,uint256)._to (CoinToken.sol#129) is not in mixedCase
Parameter StandardToken.transfer(address,uint256)._value (CoinToken.sol#129) is not in mixedCase
Parameter StandardToken.balanceOf(address)._owner (CoinToken.sol#142) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._from (CoinToken.sol#146) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._to (CoinToken.sol#146) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._value (CoinToken.sol#146) is not in mixedCase
Parameter StandardToken.approve(address,uint256)._spender (CoinToken.sol#160) is not in mixedCase
Parameter StandardToken.approve(address,uint256)._value (CoinToken.sol#160) is not in mixedCase
Parameter StandardToken.allowance(address,address)._owner (CoinToken.sol#167) is not in mixedCase
Parameter StandardToken.allowance(address,address)._spender (CoinToken.sol#167) is not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._spender (CoinToken.sol#172) is not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._addedValue (CoinToken.sol#172) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._spender (CoinToken.sol#178) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._subtractedValue (CoinToken.sol#178) is not in mixedCase
Parameter PausableToken.transfer(address,uint256)._to (CoinToken.sol#204) is not in mixedCase
Parameter PausableToken.transfer(address,uint256)._value (CoinToken.sol#204) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._from (CoinToken.sol#208) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._to (CoinToken.sol#208) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._value (CoinToken.sol#208) is not in mixedCase
Parameter PausableToken.approve(address,uint256)._spender (CoinToken.sol#212) is not in mixedCase
Parameter PausableToken.approve(address,uint256)._value (CoinToken.sol#212) is not in mixedCase
Parameter PausableToken.increaseApproval(address,uint256)._spender (CoinToken.sol#216) is not in mixedCase
Parameter PausableToken.increaseApproval(address,uint256)._addedValue (CoinToken.sol#216) is not in mixedCase
Parameter PausableToken.decreaseApproval(address,uint256)._spender (CoinToken.sol#220) is not in mixedCase
Parameter PausableToken.decreaseApproval(address,uint256)._subtractedValue (CoinToken.sol#220) is not in mixedCase
Parameter CoinToken.burn(uint256)._value (CoinToken.sol#248) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CoinToken.sol#55-59)
pause() should be declared external:
        - Pausable.pause() (CoinToken.sol#89-92)
unpause() should be declared external:
        - Pausable.unpause() (CoinToken.sol#97-100)
balanceOf(address) should be declared external:
        - ERC20Basic.balanceOf(address) (CoinToken.sol#105)
        - StandardToken.balanceOf(address) (CoinToken.sol#142-144)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (CoinToken.sol#111)
        - StandardToken.allowance(address,address) (CoinToken.sol#167-169)
blackListAddress(address,bool) should be declared external:
        - PausableToken.blackListAddress(address,bool) (CoinToken.sol#224-226)
burn(uint256) should be declared external:
        - CoinToken.burn(uint256) (CoinToken.sol#248-250)
mint(address,uint256) should be declared external:
        - CoinToken.mint(address,uint256) (CoinToken.sol#260-266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

MythX: -

```
Report for CoinToken.sol
https://dashboard.mythx.io/#/console/analyses/c6e00313-566e-4cbf-8f02-80b3dd7ddbcb
```

Line	SWC Title	Severity	Short Description
5	(SWC-103) Floating Pragma	Low	A floating pragma is set.
122	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
126	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
242	(SWC-101) Integer Overflow and Underflow	High	The arithmetic operator can overflow.

Solhint: -

Linter results:

```
CoinToken.sol:5:1: Error: Compiler version ^0.4.24 does not satisfy the r semver requirement

CoinToken.sol:89:44: Error: Visibility modifier must be first in list of modifiers

CoinToken.sol:97:43: Error: Visibility modifier must be first in list of modifiers

CoinToken.sol:122:2: Error: Explicitly mark visibility of state

CoinToken.sol:126:3: Error: Explicitly mark visibility of state

CoinToken.sol:260:62: Error: Visibility modifier must be first in list of modifiers
```

Basic Coding Bugs

No.	Name	Description	Severity	Result
1.	Constructor Mismatch	Whether the contract name and its constructor are not identical to each other.	Critical	PASSED
2.	Ownership Takeover	Whether the set owner function is not protected.	Critical	PASSED
3.	Redundant Fallback Function	Whether the contract has a redundant fallback function.	Critical	PASSED
4.	Overflows & Underflows	Whether the contract has general overflow or underflow vulnerabilities	Critical	*FOUND*
5.	Reentrancy	Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs	High	PASSED
6.	MONEY-Giving Bug	Whether the contract returns funds to an arbitrary address	High	PASSED
7.	Blackhole	Whether the contract locks ETH indefinitely: merely in without out	High	PASSED
8.	Unauthorized Self-Destruct	Whether the contract can be killed by any arbitrary address	Medium	PASSED
9.	Revert DoS	Whether the contract is vulnerable to DoS attack because	Medium	PASSED

		of unexpected revert		
10.	Unchecked External Call	Whether the contract has any external call without checking the return value	Medium	PASSED
11.	Gasless Send	Whether the contract is vulnerable to gasless send	Medium	PASSED
12.	Send Instead of Transfer	Whether the contract uses send instead of transfer	Medium	PASSED
13.	Costly Loop	Whether the contract has any costly loop which may lead to Out-Of-Gas exception	Medium	PASSED
14.	(Unsafe) Use of Untrusted Libraries	Whether the contract use any suspicious libraries	Medium	PASSED
15.	(Unsafe) Use of Predictable Variables	Whether the contract contains any randomness variable, but its value can be predicated	Medium	PASSED
16.	Transaction Ordering Dependence	Whether the final state of the contract depends on the order of the transactions	Medium	PASSED
17.	Deprecated Uses	Whether the contract use the deprecated tx.origin to perform the authorization	Medium	PASSED
18.	Semantic Consistency Checks	Whether the semantic of the white paper is different from the implementation of the contract	Critical	PASSED

Conclusion

In this audit, we thoroughly analyzed QUOTA's 'CoinToken' Smart Contract. The current code base is well organized but there are promptly some Critical and Low-level issues found in this phase of Smart Contract Audit. It's recommended to update the contract before crossing across some serious issues.

Meanwhile, we need to call attention to that smart contract as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

About Virtual Caim

Just like our other parallel journey at eNebula Solution, we believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The Virtual Caim is specifically incorporated to handle all kind of Security related operations, our Highly Qualified and Certified security team has skills for reviewing coding languages like Solidity, Rust, Go, Python, Haskell, C, C++ and JavaScript for common security vulnerabilities & specific attack vectors. The team has been reviewing implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent & open about the work we do.

For more information about our other security services and consulting,
please visit -- https://virtualcaim.com/
& Mail us at – audit@virtualcaim.com