VIDMA

Kingaru

SMART CONTRACT AUDIT

Project: Kingaru
Date: May 30, 2022

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

While conducting the smart contract audit, no issues with critical or high severity levels were detected. Audited smart contracts are following security best practices. For the best understanding and deeper dive into the codebase, Kingaru team provided the technical documentation that included detailed sequence diagrams and function descriptions.

During the audit process, the Vidma team found several issues. A detailed summary and the current state are displayed in the table below.

Severity of the issue	Total found	Resolved	Unresolved
Critical	0 issues	0 issues	0 issues
High	2 issues	2 issues	0 issues
Medium	2 issues	2 issues	0 issues
Low	3 issues	3 issues	0 issues
Informational	3 issues	3 issues	0 issues
Total	**10 issues**	**10 issues**	**0 issues**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contracts are fully operational and secure. Under the given circumstances, we set the following risk level:

Our auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and risk level of the issues are all taken into consideration. The Vidma team has developed a transparent scoring system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Based on the **overall result of the audit**, the Vidma audit team grants the following score:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level.

The test results and the coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

Kingaru

A highly scalable, fast, secure blockchain for E-Commerce, NFT, Metaverse, and Retail.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contracts.

The audit was conducted from May 7, 2022 to May 30, 2022. The outcome is disclosed in this document.

The scope of work for the given audit consists of the following contracts:

- ManagementUpgradeable;
- KRUShopsManager;
- KRUShopsPaymentProccesor;
- KRUShopsPool.

The source code was taken from the following **source**:
https://bitbucket.org/applicature/kingaru.contracts

Initial commit submitted for the audit:
f65cc3dea7cbd1907a54a8afb5182a4d3b14862b?at=audit/25.04

Last commit reviewed by the auditing team:
0edd2d85efb03a08da66b6c479e77b2314d2076d

As a reference to the contracts logic, business concept, and the expected behavior of the codebase, the Kingaru team has provided the following documentation:
https://drive.google.com/drive/folders/1oROWAGh8UtNVqwBWJnrnq1bHh-4RsPhg?usp=sharing

WORKFLOW OF THE AUDITING PROCESS

Vidma audit team uses the most sophisticated and contemporary methods and well-developed techniques to ensure contracts are free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After the Audit kick-off, our security team conducts research on the contract's logic and expected behavior of the audited contract.

Documentation reading

Vidma auditors do a deep dive into your tech documentation with the aim of discovering all the behavior patterns of your codebase and analyzing the potential audit and testing scenarios.

The outcome

At this point, the Vidma auditors are ready to kick off the process. We set the auditing strategies and methods and are prepared to conduct the first audit part.

Phase 2: Manual part of the audit

Manual check

During the manual phase of the audit, the Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract. The initial commit as stated in the agreement is taken into consideration.

Static analysis check

Static analysis tools are used to find any other vulnerabilities in smart contracts that were missed after a manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Within the testing part, Vidma auditors run integration tests using the Truffle or Hardhat testing framework. The test coverage and the test results are inserted in the accompanying section of this audit report.

The outcome

Second interim report with the list of new issues found during the testing part of the audit process.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings in accordance with the severity level of the issues. (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on if they have been fixed by Kingaru or not. The issues with “Not Valid” status are left on the list of findings but are not eligible for the score points deduction

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to 6 levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or the issue could result in a significant loss.

Example: Underflow/overflow, precisions, locked funds.

High

The issue significantly affects the ability of the contract to compile or operate. These are potential security or operational issues.

Example: Compilation errors, pausing/unpausing of some functionality, a random value, recursion, the logic that can use all gas from block (too many iterations in the loop), no limitations for locking period, cooldown, arithmetic errors which can cause underflow, etc.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: Absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but are more related to optimization of the codebase.

Example: Unused variables, inappropriate function visibility (public instead of external), useless importing of SCs, misuse or disuse of constant and immutable, absent indexing of parameters in events, absent events to track important state changes, absence of getters for important variables, usage of string as a key instead of a hash, etc.

Informational

Are classified as every point that increases onboarding time and code reading, as well as the issues which have no impact on the contract's ability to operate.

Example: Code style, NatSpec, typos, license, refactoring, naming convention (or unclear naming), layout order, functions order, lack of any type of documentation.

MANUAL REPORT

No locking period for shop commission change at KRUShopsManager.sol

High | Resolved

In functions *setGeneralCommission()* and *setSpecificCommission()* it's possible to change the commission for a shop at any point of time . After completing the order, the shop can execute the *withdraw()* function at KRUShopsPool.sol, and this will allow the shop manager to front-run shop transactions and change commissions before transaction execution. As a result, shops can receive fewer funds as expected or even all funds will be transferred to the treasury without the possibility to return to the shop.

Recommendation:

Consider adding a lock period for setting a new commission for the shop.

No possibility for BE/Shop to withdraw funds for removed, blacklisted, or frozen shop funds at KRUShopsPool.sol

High | Resolved

Functions *removeShop()*, *setShopToBlackList()*, *setShopToFreezList()* at KRUShopsManager.sol allows to shop manager to lock funds of compromised shops and disallow to withdraw funds by shop/BE even for already paid and completed orders.

Recommendation:

Add a function that allows to partially withdraw funds for orders which were already completed and paid with help of the shop manager or BE signature and available withdrawal amount as a parameter for covering the costs of orders which were completed.

Possibility of the temporary lock of funds at KRUShopsPool.sol

Medium | Resolved

In functions *setGeneralCommission()* and *setSpecificCommission()* at KRUShopsManager.sol it is possible to set commission for shops commission which can be equal to 100%. In function *withdraw()* at KRUShopsPool there is check for available amount to withdraw for shop and if in KRUShopsManager contract commission for caller shop will be equal to 100 % transaction will always fail as it will not satisfy requirement:

```
require(availToWithdraw > 0, ERROR_AMOUNT_IS_ZERO);
```

Recommendation:

Consider to specify more precise max commission in KRUShopsManager or change require statement for *setGeneralCommission()* and *setSpecificCommission()* as shown below:

```
require(commission_ < PERCENTAGE_100, ERROR_MORE_THEN_MAX);
```

Possible KRU conversion errors at KRUShopsPaymentProccesor.sol

Medium | Resolved

In the manual mode process of estimation KRU depends on BE. If any errors will occur on the BE side there is no possibility to prevent payment of orders for customers which have valid signatures. It can lead to order completion with no corresponding KRU amount paid for the order.

Recommendation:

Add pause functionality which will temporarily block new orders payment.

Insufficient BE gas refund at KRUShopsPool.sol

Low | Resolved

In function *withdrawFor()* which should be called only by BE there is functionality to refund BE for gas spent on transaction execution. But calculation not considering 2 external calls in modifier *canWithdraw()*. Currently BE will always spend more KRU for transaction execution than the amount it will receive as refund.

Recommendation:

Consider to use library AddressUpgradeable for transferring KRU to BE and shop and change *beFee* calculation as shown below:

```
uint256 beFee = tx.gasprice * (gasBefore - gasleft() + 84000);
```

Floating pragma

Low | Resolved

The current version of solc is ^0.8.11 and it is better to lock the pragma to a specific version.

Recommendation:

Lock pragma to a specific version.

Lack of zero address check

Low | Resolved

In function *registerContract()* (SC: ManagementUpgradeable) there is no check if the *target_* address isn't a zero address.

Recommendation:

Consider adding a check for zero address for function *registerContract()*.

Some kind of typos in the code source

Informational | Resolved

There is a typo *SHOPS_MAGANER_FREEZ_LIST_PERM* that occurs in smart contracts KRUShopsPool.sol and KRUShopsManager.sol

Recommendation:

Consider fixing the typo.

Typo in parameter name at ManagementUpgradeable.sol

Informational | Resolved

Function *setKycWhitelists()* use as a parameter array of addresses, but this parameter is assigned as address.

Recommendation:

Consider to change parameter name:

```
setKycWhitelists(address[] calldata address_, bool value_)
```

to

```
setKycWhitelists(address[] calldata addresses_, bool value_)
```

Confusing function and variable name at KRUShopsPaymentProccesor.sol

Informational | Resolved

Function *switchRate()* is used for switching mode of calculation of KRU price from automation to manual and vice versa.

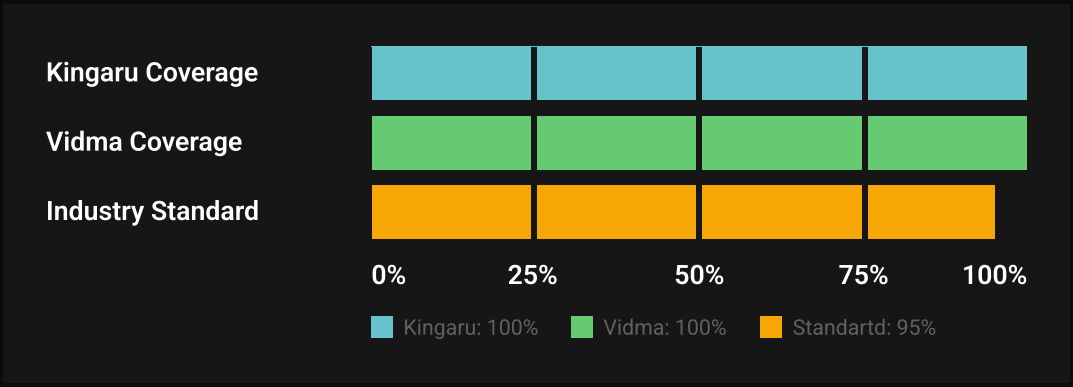
Recommendation:

Change function name from *switchRate()* to *switchMode()* and variable *isManualRate* to *isManualMode*.

TEST RESULTS

To verify the security of contracts and the performance, a number of integration tests were carried out using the Truffle testing framework.

In this section, we provide both tests written by Kingaru and tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the Kingaru repository. We write totally separate tests with code coverage of a minimum of 95% to meet the industry standards.

Tests written by Kingaru

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	98.61	100.00	100.00
KRUShopsManager.sol	100.00	100.00	100.00	100.00
KRUShopsPaymentProccesor.sol	100.00	95.00	100.00	100.00
KRUShopsPool.sol	100.00	100.00	100.00	100.00
contracts\management\	100.00	94.44	100.00	100.00
Constants.sol	100.00	100.00	100.00	100.00
ManagedUpgradeable.sol	100.00	100.00	100.00	100.00
ManagementUpgradeable.sol	100.00	90.00	100.00	100.00
All Files	**100.00**	**97.78**	**100.00**	**100.00**

Test Results

```
Contract: KRUShopsManager
  initialize
    ✓ should fail deploy KRUShopsManager if management contract
      is zero (258ms)
    ✓ should deploy KRUShopsManager successfully (211ms)
  submitChangeCommission/confirmChangeCommission
```

```
  ✓ should fail if member call submitChangeCommission without
    permission (100ms)
  ✓ should fail if commission value more than 10% (67ms)
  ✓ should fail if member call confirmChangeCommission without
    permission
  ✓ should fail if isSubmit value is false
  ✓ should set specific commission value successfully (105ms)
setRegisterMode
  ✓ should fail if member call setRegisterMode without permission
  ✓ should fail if same register mode has already setted
  ✓ should set automatic register mode successfully
setShopToBlackList
  ✓ should fail if member call setShopToBlackList without
    permission
  ✓ should fail if shop has already setted to blacklist
  ✓ should add shop to blacklist successfully
setShopToFreezeList
  ✓ should fail if member call setShopToFreezeList without
    permission
  ✓ should fail if shop has already setted to freezlist
  ✓ should add shop to freezlist successfully
registerShop
  ✓ should fail if member call registerShop without permission
  ✓ should fail if address of shop is zero
  ✓ should fail if the store has already been registered
  ✓ should register new shop manually successfully (42ms)
removeShop
  ✓ should fail if member call removeShop without permission
  ✓ should fail if shop is not exist
  ✓ should remove registered shop successfully (66ms)
checkPaymentShop
  ✓ should fail if caller checkPaymentShop is not
    KRU_SHOPS_PAYMENT_PROCCESOR SC
  ✓ should fail if address of shop is zero
  ✓ should fail if shop in blacklist
  ✓ should be nothing if shop was register before and not in
    blacklist
  ✓ should fail if disable regime with automatically register
  ✓ should register new shop with auto mode successfully (59ms)
getCommission
  ✓ should fail if shop is not exist
  ✓ should return commission value for shop successfully (81ms)
```

```
  isRegistered
    ✓ should return false(if was not register), otherwise -
      true (40ms)
  Test case
    ✓ manual register flow (271ms)
    ✓ auto register flow (167ms)

Contract: KRUShopsPaymentProccesor
  Functions
    pay
      ✓ Should pay KRU correctly (58ms)
      Pay fulfilled order
        ✓ Should revert if order is already paid (59ms)
        Pay new order
          ✓ Should fail if usdAmount is zero
          ✓ Should fail if kru amount is not enough
          ✓ Should fail if expire date is less than current
            timestamp
          ✓ Should fail if expire date is less than current
            timestamp
          ✓ Should fail if incorrect parameter was provided to
            signature
          ✓ Should pay new order correctly (86ms)
      setRate
        ✓ Should set rate correctly
        ✓ Should fail if rate is zero
      switchMode
        ✓ Should turn on rate mode correctly
        ✓ Should turn off rate mode correctly
      getEstimatedKRU
        ✓ Should get estimated KRU by manual rate
        ✓ Should get estimated KRU by uniswap rate

Contract: KRUShopsPool
  Functions
    orderIn
      ✓ Should order in shop correctly (47ms)
      ✓ Should fail order in shop if shop has zero address
    withdraw
      ✓ Should fail if shop is not registered
      ✓ Should fail if shop is not registered
      ✓ Should withdraw funds correctly (102ms)
```

```
      ✓ Should fail if usdAmount is zero
    setWithdrawalCooldown
      ✓ Should fail if it was setted
      ✓ Should set withdrawal cooldown correctly
    withdrawFor flows
      ✓ Should fail if automatic flow is not executed
      ✓ Should withdraw for shop correctly for the first time (97ms)
      ✓ Should fail if cooldown period is not finished (61ms)
      ✓ Should correctly withdraw after cooldown (43ms)
      ✓ Should fail if zero usdAmount
    withdrawFromCompromised
      ✓ Should fail if shop is not compromised
      ✓ Should correctly withdraw amount (49ms)
      fails inside withdraw
        ✓ Should revert if time out
        ✓ Should revert if nonce is invalid
        ✓ Should revert if amount is bigger than balance

Contract: ManagedUpgradeable
  Deploy
    ✓ Should initialize field correct
    ✓ Should fail if try initialize secondly
    ✓ Should fail if try initialize zero address (56ms)
  Functions
    setManagementContract
      ✓ Should fail if caller not owner
      ✓ Should fail if try setup zero_address
      ✓ Should correct setup Management
    hasPermission
      ✓ Should fail if caller haven't permissions
      ✓ Should correct return
    canCallOnly
      ✓ Should correct return
      ✓ Should fail if caller call from not registered contract

Contract: ManagementUpgradeable
  Functions
    registerContract
      ✓ Should fail if users can owner
      ✓ Should correct register contract
    permissions
      ✓ Should return correct if permissions user have
```

```
    setLimitSetPermission
      ✓ Should fail if users can owner
      ✓ Should correct set limit permission
      ✓ Should correct set limit permission - false
    setPermissions
      ✓ Should fail if users can owner
      ✓ Should correct set permissions
    setPermission
      ✓ Should fail if users can owner
      ✓ Should correct set permissions
      ✓ Should correct set permission from not owner but with limit
        set permissions true
      ✓ Should fail if try call from user with limit but with
        another permissions which allow
    setKycWhitelists
      ✓ Should fail if users can have
        MANAGEMENT_CAN_SET_KYC_WHITELISTED
      ✓ Should correct set kyc whitelisted
    isKYCPassed
      ✓ fail if time out of sign
      ✓ return true if addres have MANAGEMENT_WHITELISTED_KYC
      ✓ return true if user correct push sign
      ✓ fail if user correct push invalid sign
      ✓ fail if time out of sign

  95 passing (20s)
```

Tests written by Vidma auditors

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
KRUShopsManager.sol	100.00	100.00	100.00	100.00
KRUShopsPaymentProccesor.sol	100.00	100.00	100.00	100.00
KRUShopsPool.sol	100.00	100.00	100.00	100.00
contracts\management\	100.00	100.00	100.00	100.00
Constants.sol	100.00	100.00	100.00	100.00
ManagedUpgradeable.sol	100.00	100.00	100.00	100.00
ManagementUpgradeable.sol	100.00	100.00	100.00	100.00
All Files	**100.00**	**100.00**	**100.00**	**100.00**

Test Results

```
Contract: KRUShopsManager
  KRUShopsManager Test Cases
    KRUShopsManager Initialize Test Cases
      ✓ shouldn't initialize with management setted to the zero
        address (54ms)
      ✓ should initialize with correct owner (69ms)
```

```
  ✓ should initialize with correct management
  ✓ hould initialize with correct initial general commission
  ✓ shouldn't initialize if already initialized
KRUShopsManager Shop Registration/Removal Test Cases
  ✓ shouldn't allow to register a shop without proper authority
  ✓ shouldn't allow to register a shop with zero address (40ms)
  ✓ should register a shop correctly (215ms)
  ✓ shouldn't register a shop if already registered
  ✓ should check if shop registered correctly
  ✓ shouldn't allow to remove a shop without proper authority
  ✓ shouldn't allow to remove not existing shop
  ✓ should remove a shop correctly (60ms)
KRUShopsManager Shop Commission Test Cases
  KRUShopsManager Shop Submitting Commission Test Cases
    ✓ shouldn't allow to submit general commission for shops
      without proper authority
    ✓ shouldn't allow to submit general commission greater
      than max commission
    ✓ should submit general commission correctly (50ms)
    ✓ should submit specific commission for shop
      correctly (46ms)
  KRUShopsManager Shop Confirming Commission Test Cases
    ✓ shouldn't change commission if not confirmed
    ✓ shouldn't allow to confirm commission for shops without
      proper authority
    ✓ shouldn't allow to confirm commission for shops if
      commision is not submitted
    ✓ shouldn't allow to confirm commission before delay (41ms)
    ✓ should confirm commission correctly (96ms)
  KRUShopsManager Shop View Commission Test Cases
    ✓ shouldn't return commission for not existing shop
    ✓ should return general commission if specific commission
      not settled for chosen shop
    ✓ should return specific commission if specific commission
      is settled for chosen shop
  KRUShopsManager Shop BlackList Test Cases
    ✓ shouldn't allow to add shop to BlackList without proper
      authority
    ✓ shouldn't allow to change shop BlackList status to
      current shop BlackList status (38ms)
    ✓ should allow to add shop to BlackList correctly (57ms)
    ✓ should allow to remove shop to BlackList correctly (91ms)
```

```
    KRUShopsManager Shop FreezeList Test Cases
      ✓ shouldn't allow to add shop to FreezeList without proper
        authority
      ✓ shouldn't allow to change shop FreezeList status to
        current shop FreezeList status
      ✓ should allow to remove shop to FreezeList correctly (54ms)
      ✓ should allow to add shop to FreezeList correctly (61ms)
    KRUShopsManager Register Mode Test Cases
      ✓ shouldn't allow to change register mode without proper
        authority
      ✓ shouldn't allow to set register mode to current shop
        manager mode
      ✓ should set register mode to automatic correctly (59ms)
      ✓ should set register mode to manual correctly (44ms)
    KRUShopsManager Shop Payment Test Cases
      ✓ shouldn't allow to check payment without proper contract
        caller authority
      ✓ shouldn't allow to check payment for zero address
      ✓ shouldn't allow to check payment for shop in
        BlackLis (38ms)
      ✓ should check payment for shop correctly
      ✓ shouldn't allow to check payment and register new shop
        in manual mode
      ✓ shouldn't check payment and register new shop in
        automatic mode (85ms)

Contract: KRUShopsPaymentProccesor
  KRUShopsPaymentProccesor Test Cases
    KRUShopsPaymentProccesor Initialize Test Cases
      ✓ shouldn't initialize with management setted to the zero
        address
      ✓ should initialize with correct owner (44ms)
      ✓ should initialize with correct management
      ✓ should initialize with correct initial rate mode
      ✓ should initialize with correct initial pause state
      ✓ shouldn't initialize if already initialized
    KRUShopsPaymentProccesor Set Rate Test Cases
      ✓ shouldn't allow to set new rate without proper authority
      ✓ shouldn't allow to set new rate equal to 0
      ✓ should set new rate correctly (42ms)
    KRUShopsPaymentProccesor Switch Mode Test Cases
      ✓ shouldn't allow to set new mode without proper authority
```

```
      ✓ should set new mode correctly (42ms)
    KRUShopsPaymentProccesor Pause Test Cases
      ✓ shouldn't allow to pause without proper authority
      ✓ shouldn't allow to set new mode without proper
        authority (40ms)
    KRUShopsPaymentProccesor KRU Estimation Process Test Cases
      ✓ should return correct KRU needed in manual mode
      ✓ shouldn't return KRU needed in automatic mode if reserves
        are empty (68ms)
      ✓ should return correct KRU needed in automatic mode (860ms)
    KRUShopsPaymentProccesor Order Pay Test Cases
      ✓ shouldn't allow to paid for order when paused (72ms)
      ✓  shouldn't allow to paid for order with zero amount of usd
      ✓ shouldn't allow to paid for order with insufficient amount
        of KRU (51ms)
      ✓ shouldn't allow to paid for order with expired
        deadline (80ms)
      ✓ shouldn't allow to paid for order with invalid
        signature (108ms)
      ✓ shouldn't allow to paid for order if shop is in
        blacklist (124ms)
      ✓ should allow to paid for order correctly (175ms)
      ✓ shouldn't allow to paid for order if shop is not registered
        and shop manager registration mode is manual (135ms)
      ✓ shouldn't allow to paid one order twice (61ms)
      ✓ should allow to paid for order if shop is not registered
        and shop manager registration mode is automatic (201ms)

Contract: KRUShopsPool
  KRUShopsPool Test Cases
    KRUShopsPool Initialize Test Cases
      ✓ shouldn't initialize with management setted to the zero
        address
      ✓ should initialize with correct owner (41ms)
      ✓ should initialize with correct management
      ✓ should initialize with correct cooldown periods (63ms)
      ✓ shouldn't initialize if already initialized
    KRUShopsPool Order Test Cases
      ✓ shouldn't place order by not the shop payment processor
      ✓ should place order correctly (49ms)
    KRUShopsPool Withdrawal Cooldown Test Cases
```

```
    ✓ shouldn't allow to set withdrawal cooldown to not
      registered shop
    ✓ shouldn't allow to set withdrawal cooldown to shop from
      blacklist
    ✓ shouldn't allow to set withdrawal cooldown to shop from
      freezlist (82ms)
    ✓ shouldn't allow to set withdrawal cooldown to current
      withdrawal cooldown (40ms)
    ✓ should set withdrawal cooldown correctly (49ms)
  KRUShopsPool Shop Withdrawal Test Cases
    ✓ shouldn't allow to withdraw to not registered shop
    ✓ shouldn't allow to withdraw to shop from blacklist
    ✓ shouldn't allow to withdraw to shop from freezlist (116ms)
    ✓ shouldn't allow to withdraw if there is no available
      amount to withdraw (48ms)
    ✓ should withdraw correctly if general commission is applied
      for shop (140ms)
    ✓ should withdraw correctly if specific commission is applied
      for shop (214ms)
  KRUShopsPool BE Withdrawal Test Cases
    ✓ shouldn't allow to withdraw for shop without proper
      authority
    ✓ shouldn't allow to withdraw for shop if shop is not
      registered
    ✓ shouldn't allow to withdraw for shop from blacklist (38ms)
    ✓ shouldn't allow to withdraw for shop from freezlist (118ms)
    ✓ shouldn't allow to withdraw for shop if there is no
      available amount to withdraw (74ms)
    ✓ shouldn't allow to withdraw for shop if cooldown is not
      setted (47ms)
    ✓ should withdraw for shop correctly (204ms)
    ✓ shouldn't withdraw for shop if shop is in cooldown period
      period (51ms)
  KRUShopsPool Compromised Shop Withdrawal Test Cases
    ✓ shouldn't withdraw compromised funds without proper
      authority (56ms)
    ✓ shouldn't withdraw compromised funds with expired
      deadline (59ms)
    ✓ shouldn't withdraw compromised funds if shop not in
      compromised list (78ms)
    ✓ shouldn't withdraw compromissed funds if shop don't have
      available amount to withdraw (115ms)
```

```
      ✓ should withdraw compromised funds correctly (224ms)
      ✓ should withdraw compromised funds with used nonce (196ms)

Contract: ManagedUpgradeable
  ManagedUpgradeable Test Cases
    Managed Initialize Test Cases
      ✓ shouldn't initialize with management setted to the zero
        address
      ✓ should initialize with correct owner
      ✓ should initialize with correct management
      ✓ shouldn't initialize if already initialized
    Managed Owner Test Cases
      ✓ shouldn't set management contract by not the current owner
      ✓ shouldn't set management contract to zero address
      ✓ should set new management contract correctly (54ms)
    Managed Permission Test Cases
      ✓ shouldn't allow to execute function if caller hasn't
        permission
      ✓ should return false if subject don't have required
        permission
      ✓ should return true if subject have required permission (46ms)
    Managed Registered Contracts Test Cases
      ✓ shouldn't allow to call for not registered contract
      ✓ should allow to call for registered contract

Contract: ManagementUpgradeable
  ManagementUpgradeable Test Cases
    Management Owner Settings Test Cases
      ✓ shouldn't grant limit permission by not the current owner
      ✓ should grant limit permission correctly (38ms)
      ✓ shouldn't set permission by not the current owner or
        address without permission grant rights
      ✓ should set permission by the current owner correctly
      ✓ shouldn't set multiple permissions by not the current owner
      ✓ should set multiple permissions by the current owner
        correctly
      ✓ shouldn't register contract by not the current owner
      ✓ shouldn't register contract with zero address
      ✓ should register contract correctly
    Management Limit Permission Test Cases
      ✓ shouldn't set permission by the address without permission
        grant rights
```

```
    ✓ should set permission by the address with permission grant
      rights correctly
  Management KYC Whitelist Test Cases
    ✓ shouldn't set whitelisted KYC by not the management of KYC
      whitelist
    ✓ should set whitelisted KYC correctly
    ✓ shouldn't pass KYC with expired deadline
    ✓ shouldn't pass KYC if address not whitelisted and with not
      valid signature
    ✓ should pass KYC verification for whitelisted addresses
    ✓ should pass KYC verification with valid signature

130 passing (12s)
```

We are delighted to have a chance to work with the Kingaru team and contribute to your company's success by reviewing and certifying the security of your smart contracts.

The statements made in this document should be interpreted neither as investment or legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists. Our company is based in Ukraine, known for its strong engineering, cryptography, and cybersecurity culture.

Website: **vidma.io**
Email: **security@vidma.io**

VIDMA