Quantstamp DRAFT

Kim Dex

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	DEX
Timeline	2023-12-19 through 2023-12-21
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Diff/Fork information	As this is a diff audit, below is the fork information of each of the forked repositories: 1. `kizuna-contracts` ○ **Original repository**: https://github.com/CamelotLabs/core ○ **Fork commit hash**: 91f95e0f67d6424a76ac2f53ee29b68b51ab319a 2. `kizuna-router` ○ **Original repository**: https://github.com/CamelotLabs/periphery ○ **Fork commit hash**: 132e3627d153e463575da207b00c62214b954c2a 3. `kizuna-tokens` ○ **Original deployed contract**: https://arbiscan.io/address/0x3d9907F9a368ad0a51Be60f7Da3b97cf940982D8
Source Code	• https://github.com/kizuna-dex/kizuna-contracts #1cfea3c • https://github.com/kizuna-dex/kizuna-router #09d7258 • kizuna-dex/kizuna-tokens #fe035b8
Auditors	• Mustafa Hasan Senior Auditing Engineer • Michael Boyle Auditing Engineer • Rabib Islam Auditing Engineer

Documentation quality	Low
Test quality	Low
Total Findings	8 Fixed: 1 Acknowledged: 6 Mitigated: 1
High severity findings ⓘ	1 Fixed: 1
Medium severity findings ⓘ	0
Low severity findings ⓘ	3 Acknowledged: 3
Undetermined severity findings ⓘ	0
Informational findings ⓘ	4 Acknowledged: 3 Mitigated: 1

Summary of Findings

Kim DEX is a fork of the Camelot DEX project, which will be deployed on the Mode L2 blockchain. One of Mode's features is the Sequencer Fee Sharing mechanism that allows contract owners to register their contracts in a fee sharing pool, enabling them to earn a portion of the fees generated by the contracts. The scope of this diff audit is Kim's implementation of the registration, assignment, and withdrawal logic on top of the forked Camelot DEX codebase.

Overall, the code is well written and inline documentation is fair, but branch coverage for the test suite is very poor for the three repositories in-scope for the audit.

The audit yielded a high severity issue where any user could withdraw SFS and call arbitrary contracts, as well as a number of low severity and informational issues.

Fix Review: The team fixed the high severity issue and acknowledged most of the rest of the findings. Among the changes made to the codebases was the renaming of the repositories so their prefixes are `kim` instead of `kizuna`.

ID	DESCRIPTION	SEVERITY	STATUS
KIM-1	Lack of Access Control Allows Withdrawal to Any Address	High	Fixed
KIM-2	Ownership Can Be Renounced	Low	Acknowledged
KIM-3	Outdated Solidity Compiler Version	Low	Acknowledged
KIM-4	Unlocked Pragma	Low	Acknowledged
KIM-5	`feeSharingContract` Address Not Hardcoded	Informational	Acknowledged
KIM-6	Unnecessarily Complicated External Calls	Informational	Acknowledged
KIM-7	Not Registering on Deployment	Informational	Acknowledged
KIM-8	Unnecessary Implementation of `withdraw()`	Informational	Mitigated

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope of the audit included the implementation of the `register()`, `assign()` and `withdraw()` functions in the following files:

- kizuna-contracts/contracts/KizunaFactory.sol
- kizuna-contracts/contracts/KizunaPair.sol
- kizuna-router/contracts/KizunaRouter.sol
- kizuna-tokens/contracts/RyoToken.sol
- kizuna-tokens/contracts/XRyoToken.sol

Findings

KIM-1 Lack of Access Control Allows Withdrawal to Any Address

● High ⓘ Fixed

✓ **Update**

Marked as "Fixed" by the client. Addressed in: `45f3059cb6308f74e32837e4c119e837cccf6548`. The client provided the following explanation:

> Fixed by adding owner only. Repo is actually: Repo is actually: https://github.com/kizuna-dex/kizuna-router Latest router is 96c556fd9a94e3055286fd26f8d25249d94e4327 just a rename from KizunaRouter to KimRouter

File(s) affected: `kizuna-router/contracts/KizunaRouter.sol`

Description: The `register()`, `assign()` and `withdraw()` functions in the `KizunaRouter.sol` are missing an access modifier to prevent them from being publicly callable by anyone. Of the three functions, `withdraw()` bears the most impact since it allows an attacker to supply an arbitrary `recipient` address to withdraw the contract's fee share to, effectively stealing funds.

This also opens up the possibility to perform function calls on arbitrary external contracts since the `feeSharingContract` is user supplied to all three functions, which could allow an attacker to call functions with the same signature on other contracts with the `msg.sender` being the `KizunaRouter` contract's address, leading to unwanted behavior or other, possibly more impactful consequences.

Recommendation: Add modifiers to the three functions so only authorized users can invoke them.

KIM-2 Ownership Can Be Renounced

● Low ⓘ Acknowledged

ⓘ **Update**

Marked as "Acknowledged" by the client. The client provided the following explanation:

> Will update documentation to make this more clear

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol`, `kizuna-contracts/contracts/KizunaPair.sol`, `kizuna-tokens/contracts/RyoToken.sol`, `kizuna-tokens/contracts/XRyoToken.sol`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

Recommendation: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

KIM-3 Outdated Solidity Compiler Version

● Low ⓘ Acknowledged

ⓘ **Update**

Marked as "Acknowledged" by the client. The client provided the following explanation:

> Solidity version being dated is not an issue and also relevant in forked version of repo.

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol`, `kizuna-contracts/contracts/KizunaPair.sol`, `kizuna-router/contracts/KizunaRouter.sol`, `kizuna-tokens/contracts/RyoToken.sol`, `kizuna-tokens/contracts/XRyoToken.sol`

Description: As security standards develop, so does the Solidity language. In order to stay up to date with current practices, it's important to use a recent version of Solidity.

Recommendation: Upgrade the solidity version to `0.8.18`.

KIM-4 Unlocked Pragma

• Low Acknowledged

> **Update**
>
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > Not a concern for locking the pragma version at this time.

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol`, `kizuna-contracts/contracts/KizunaPair.sol`, `kizuna-router/contracts/KizunaRouter.sol`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, we recommend to remove the caret to lock the file onto a specific Solidity version.

KIM-5 `feeSharingContract` Address Not Hardcoded

• Informational Acknowledged

> **Update**
>
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > Not relevant but will update documentation to be more clear

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol`, `kizuna-contracts/contracts/KizunaPair.sol`, `kizuna-router/contracts/KizunaRouter.sol`, `kizuna-tokens/contracts/RyoToken.sol`, `kizuna-tokens/contracts/XRyoToken.sol`

Description: The `register()`, `assign()`, and `withdraw()` functions are only intended to call a specific instance of the fee-sharing contract. However, the caller can specify an arbitrary address to make calls to. This could be potentially dangerous if another contract uses the same function name and parameters.

Recommendation: Assuming the contract address should never change, consider making the address of the `feeSharingContract` immutable and set it during the deployment of the contract.

KIM-6 Unnecessarily Complicated External Calls

• Informational Acknowledged

> **Update**
>
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > Low risk, contract was audited and will manage through internal dev process

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol`, `kizuna-contracts/contracts/KizunaPair.sol`, `kizuna-router/contracts/KizunaRouter.sol`, `kizuna-tokens/contracts/RyoToken.sol`, `kizuna-tokens/contracts/XRyoToken.sol`

Description: Currently, low-level calls are being used to interact with the `FeeSharing` contract. However, this is unnecessary: an interface for `FeeSharing` can be written with the functions `register()`, `assign()`, and `withdraw()`. This interface can then be used to call the corresponding functions on the `FeeSharing` contract instead of using low-level calls.

Recommendation: Consider writing an interface for `FeeSharing` and using that instead of using low-level calls.

KIM-7 Not Registering on Deployment

• Informational Acknowledged

> **Update**

> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > This is manually done through internal business process. Not concerned with registering contract on deployment

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol` , `kizuna-contracts/contracts/KizunaPair.sol` , `kizuna-router/contracts/KizunaRouter.sol` , `kizuna-tokens/contracts/RyoToken.sol` , `kizuna-tokens/contracts/XRyoToken.sol`

Description: Supposing that a contract starts earning fees once it is registered, it would be in the best interest of a contract owner to register their contract as soon as it is deployed. Additionally, as per the documentation in `FeeSharing.sol` , "every contract is responsible to register itself in the constructor by calling `register(address)` ."

Recommendation: Consider calling `FeeSharing.register()` in the constructor of contracts that are to be registered.

KIM-8 Unnecessary Implementation of `withdraw()`

● Informational ⓘ Mitigated

> **Update**
>
> Marked as "Fixed" by the client. Addressed in: `2bd33229a74c590c271ce87471d4391e69fd7a0e` . The client provided the following explanation:
>
> > Issue is fixed Repo is actually: https://github.com/kizuna-dex/kizuna-router
>
> **Fix Review:** The function was only removed from `kizuna-router` and is still kept for `kizuna-tokens` and `kizuna-contracts` .

File(s) affected: `kizuna-contracts/contracts/KizunaFactory.sol` , `kizuna-contracts/contracts/KizunaPair.sol` , `kizuna-router/contracts/KizunaRouter.sol` , `kizuna-tokens/contracts/RyoToken.sol` , `kizuna-tokens/contracts/XRyoToken.sol`

Description: The affected contracts are implementing `withdraw()` . However, the `withdraw()` function is to be called by the owner of `FeeSharing` NFTs. Therefore, unless these contracts are designed to be receiving NFTs (which does not appear to be obviously the case due to not implementing the `ERC721TokenReceiver` interface), they have no need to implement `withdraw()` as they currently do.

Recommendation: Consider removing the `withdraw()` function from the affected contracts.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

- `3be...6a4 ./KizunaRouter.sol`
- `368...ecd ./Migrations.sol`
- `10b...ca3 ./UniswapV2Library.sol`

- `f67...244 ./SafeMath.sol`
- `ac1...a29 ./IKizunaRouter.sol`
- `6db...37e ./IWETH.sol`
- `66a...499 ./IUniswapV2Router01.sol`
- `6d3...3aa ./ERC20.sol`
- `947...58a ./RouterEventEmitter.sol`
- `2a8...12a ./WETH9.sol`
- `ef1...ffa ./DeflatingERC20.sol`
- `9bd...aab ./RyoToken.sol`
- `652...d75 ./XRyoToken.sol`
- `94a...2b1 ./IXRyoTokenUsage.sol`
- `e30...475 ./IXRyoToken.sol`
- `dc1...c8f ./IRyoToken.sol`
- `d86...aa0 ./contracts/KizunaFactory.sol`
- `831...d24 ./contracts/KizunaPair.sol`
- `368...ecd ./contracts/Migrations.sol`
- `70c...f8e ./contracts/UniswapV2ERC20.sol`
- `1e5...eb1 ./contracts/libraries/UQ112x112.sol`
- `807...ce6 ./contracts/libraries/Math.sol`
- `79c...6c9 ./contracts/libraries/SafeMath.sol`
- `84e...c41 ./contracts/interfaces/IKizunaFactory.sol`
- `183...d47 ./contracts/interfaces/IKizunaPair.sol`
- `e8d...8e8 ./contracts/interfaces/IUniswapV2Callee.sol`
- `837...690 ./contracts/interfaces/IUniswapV2ERC20.sol`
- `2b2...cf5 ./contracts/interfaces/IERC20.sol`
- `a87...b17 ./contracts/test/ERC20.sol`
- `165...552 ./contracts/test/WETH.sol`

Tests

- `e20...244 ./UniswapV2Router02.spec.ts`
- `738...cec ./fixtures.ts`
- `b19...514 ./utilities.ts`
- `529...093 ./UniswapV2Pair.spec.ts`
- `482...af8 ./UniswapV2Factory.spec.ts`
- `55d...477 ./StableSwap.spec.ts`
- `fed...be8 ./UniswapV2ERC20.spec.ts`
- `44a...6bd ./fixtures.ts`
- `822...c68 ./utilities.ts`
- `220...536 ./test/FeeSharing.t.sol`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

All results were either included as findings in the report, out of scope for the audit, or false positives that were discarded.

Test Suite Results

It is worth noting that neither of the three repositories has coverage for in-scope code. That is, none of the existing tests covers the three functions that are in-scope for this audit.

Fix Review: Tests were added to `kim-tokens`, which previously had no test suite present. We would like to emphasize that coverage for the three functions in-scope for this audit remains lacking.

- `kim-contracts`:

```
StableSwap
    ✔ mint (53ms)
    ✔ getInputPrice:0 (235ms)
    ✔ getInputPrice:1 (181ms)
    ✔ getInputPrice:2 (180ms)
    ✔ getInputPrice:3 (110ms)
    ✔ getInputPrice:4 (101ms)
    ✔ getInputPrice:5 (106ms)
    ✔ getInputPrice:6 (157ms)
    ✔ getInputPrice:7 (176ms)
    ✔ getInputPrice:8 (165ms)
    ✔ getInputPrice:9 (133ms)
    ✔ getInputPrice:10 (116ms)
    ✔ getInputPrice:11 (109ms)
    ✔ getInputPrice:12 (173ms)
    ✔ getInputPrice:13 (156ms)
    ✔ getInputPrice:14 (145ms)
    ✔ getInputPrice:15 (129ms)
    ✔ getInputPrice:16 (97ms)
    ✔ getInputPrice:17 (96ms)
    ✔ getInputPrice:18 (93ms)
    ✔ getInputPrice:19 (153ms)
    ✔ getInputPrice:20 (152ms)
    ✔ getInputPrice:21 (158ms)
    ✔ optimistic:0 (131ms)
    ✔ optimistic:1 (94ms)
    ✔ optimistic:2 (114ms)
    ✔ optimistic:3 (85ms)
    ✔ optimistic:4 (120ms)
    ✔ optimistic:5 (139ms)
    ✔ optimistic:6 (135ms)
    ✔ optimistic:7 (132ms)
    ✔ optimistic:8 (94ms)
    ✔ optimistic:9 (84ms)
    ✔ optimistic:10 (85ms)
    ✔ optimistic:11 (84ms)
    ✔ swap:token0 (119ms)
    ✔ swap:token1 (138ms)
    ✔ burn (117ms)
    ✔ feeTo:off (145ms)
    ✔ feeTo:on (114ms)
    From UNI to Stable
      ✔ mint (69ms)
      ✔ burn (107ms)
      ✔ swap (125ms)

  UniswapV2ERC20
    ✔ name, symbol, decimals, totalSupply, balanceOf, DOMAIN_SEPARATOR, PERMIT_TYPEHASH
    ✔ approve
    ✔ transfer
    ✔ transfer:fail
    ✔ transferFrom
    ✔ transferFrom:max

  KimFactory
    ✔ feeTo, allPairsLength
    ✔ createPair (65ms)
    ✔ createPair:reverse (70ms)
    ✔ createPair:gas
    ✔ setFeeTo

  KimPair
    ✔ mint (65ms)
    ✔ getInputPrice:0 (80ms)
    ✔ getInputPrice:1 (74ms)
    ✔ getInputPrice:2 (74ms)
    ✔ getInputPrice:3 (74ms)
    ✔ getInputPrice:4 (102ms)
    ✔ getInputPrice:5 (116ms)
    ✔ getInputPrice:6 (120ms)
    ✔ getInputPrice:7 (126ms)
    ✔ getInputPrice:8 (130ms)
    ✔ getInputPrice:9 (100ms)
    ✔ getInputPrice:10 (74ms)
    ✔ getInputPrice:11 (99ms)
    ✔ getInputPrice:12 (79ms)
    ✔ getInputPrice:13 (77ms)
    ✔ getInputPrice:14 (116ms)
    ✔ getInputPrice:15 (128ms)
    ✔ getInputPrice:16 (116ms)
    ✔ getInputPrice:17 (116ms)
    ✔ getInputPrice:18 (118ms)
    ✔ getInputPrice:19 (82ms)
    ✔ getInputPrice:20 (78ms)
    ✔ optimistic:0 (72ms)
    ✔ optimistic:1 (70ms)
    ✔ optimistic:2 (70ms)
    ✔ optimistic:3 (108ms)
    ✔ optimistic:4 (108ms)
    ✔ optimistic:5 (104ms)
    ✔ optimistic:6 (116ms)
    ✔ optimistic:7 (117ms)
    ✔ optimistic:8 (90ms)
    ✔ optimistic:9 (72ms)
    ✔ optimistic:10 (92ms)
    ✔ optimistic:11 (81ms)
    ✔ swap:token0 (74ms)
    ✔ swap:token1 (101ms)
    ✔ burn (133ms)
    ✔ feeTo:off (135ms)
    ✔ feeTo:on (186ms)


  93 passing (11s)
```

- `kim-router`:

```
UniswapV2Router02
    ✔ quote
    ✔ getAmountsOut
    fee-on-transfer tokens
      ✔ removeLiquidityETHSupportingFeeOnTransferTokens (64ms)
      swapExactTokensForTokensSupportingFeeOnTransferTokens
        ✔ DTT -> WETH
        ✔ WETH -> DTT
    fee-on-transfer tokens: reloaded
      swapExactTokensForTokensSupportingFeeOnTransferTokens
        ✔ DTT -> DTT2


  6 passing (26s)
```

- `kim-tokens`

```
KimToken
    ✔ basic info
    ✔ init master
    ✔ init emission start
    ✔ update allocations
    ✔ update emission rate
```

```
  ✔ update max supply
  ✔ update treasury address
  ✔ emit allocations
  ✔ claim master rewards

XKimToken
  ✔ basic info
  ✔ update redeem settings
  ✔ update dividends address
  ✔ update deallocation fee
  ✔ update transfer whitelist
  ✔ approve usage
  ✔ convert
  ✔ redeem (55ms)
  ✔ cancel redeem (46ms)
  ✔ update redeem dividends address


19 passing (1s)
```

Code Coverage

It was found that coverage for the three repositories is very poor, with coverage being `47.06%` for `kizuna-contracts`, `35.45%` for `kizuna-router`, and even `0%` for `kizuna-tokens`. We recommend that the tests are improved and implemented to fix the errors and cover at least `90%` of the code for all the projects.

Fix review: Coverage remains low for the three repositories, with `48.7%` and `36.11%`coverage for `kizuna-contracts` and `kizuna-router`, respectively. As for `kizuna-tokens`, which initially had no test suite present, coverage was slightly improved to `35.63%`.

- `kizuna-contracts`:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	75.25	50	64.15	76.68	
KimFactory.sol	58.82	30.43	50	62.75	... 193,197,199
KimPair.sol	79.73	57.89	67.86	80.5	... 676,680,682
Migrations.sol	0	0	0	0	9,13,17
UniswapV2ERC20.sol	73.68	33.33	88.89	82.76	... 117,118,122
contracts/interfaces/	100	100	100	100	
IERC20.sol	100	100	100	100	
IFeeSharing.sol	100	100	100	100	
IKimFactory.sol	100	100	100	100	
IKimPair.sol	100	100	100	100	
IUniswapV2Callee.sol	100	100	100	100	
IUniswapV2ERC20.sol	100	100	100	100	
contracts/libraries/	85.71	57.14	71.43	76.92	
Math.sol	75	33.33	100	87.5	21
SafeMath.sol	100	75	100	100	
UQ112×112.sol	100	100	0	0	14,19
contracts/test/	6.67	0	12.5	4.76	
ERC20.sol	100	100	100	100	
WETH.sol	0	0	0	0	... 58,59,61,63
All files	70.98	48.7	58.82	71.92	

- `kizuna-router`:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	50	27.27	50	52.53	
KimRouter.sol	50.67	29.03	55	54.17	... 450,461,472
Migrations.sol	0	0	0	0	9,13,17
contracts/interfaces/	100	100	100	100	
IFeeSharing.sol	100	100	100	100	
IKimRouter.sol	100	100	100	100	
IUniswapV2Router01.sol	100	100	100	100	
IWETH.sol	100	100	100	100	
contracts/libraries/	100	72.73	100	100	
SafeMath.sol	100	50	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
UniswapV2Library.sol	100	85.71	100	100	
contracts/test/	58.73	25	67.86	65.17	
DeflatingERC20.sol	76.19	16.67	88.89	80.65	... 120,121,125
ERC20.sol	50	16.67	60	56.67	... 122,123,127
RouterEventEmitter.sol	0	100	0	0	... 58,69,83,84
WETH9.sol	84.62	37.5	83.33	84.21	47,68,69
All files	57.89	36.11	65.52	62.5	

- `kizuna-tokens`:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	0	0	0	0	
RyoToken.sol	0	0	0	0	... 396,398,410
XRyoToken.sol	0	0	0	0	... 813,824,837
contracts/interfaces/	100	100	100	100	
IXRyoTokenUsage.sol	100	100	100	100	
contracts/interfaces/tokens/	100	100	100	100	
IRyoToken.sol	100	100	100	100	
IXRyoToken.sol	100	100	100	100	
All files	0	0	0	0	

Changelog

- 2023-12-21 - Initial report
- 2024-01-12 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp

© 2024 – Quantstamp, Inc.