CAIRO SECURITY CLAN

LAYERAKIRA

SECURITY ASSESEMENT REPORT

MAY 2025

Prepared for

LAYERAKIRA

Contents

1 About Cairo Security Clan

Cairo Security Clan is a leading force in the realm of blockchain security, dedicated to fortifying the foundations of the digital age. As pioneers in the field, we specialize in conducting meticulous smart contract security audits, ensuring the integrity and reliability of decentralized applications built on blockchain technology.

At Cairo Security Clan, we boast a multidisciplinary team of seasoned professionals proficient in blockchain security, cryptography, and software engineering. With a firm commitment to excellence, our experts delve into every aspect of the Web3 ecosystem, from foundational layer protocols to application-layer development. Our comprehensive suite of services encompasses smart contract audits, formal verification, and real-time monitoring, offering unparalleled protection against potential vulnerabilities.

Our team comprises industry veterans and scholars with extensive academic backgrounds and practical experience. Armed with advanced methodologies and cutting-edge tools, we scrutinize and analyze complex smart contracts with precision and rigor. Our track record speaks volumes, with a plethora of published research papers and citations, demonstrating our unwavering dedication to advancing the field of blockchain security.

At Cairo Security Clan, we prioritize collaboration and transparency, fostering meaningful partnerships with our clients. We believe in a customer-oriented approach, engaging stakeholders at every stage of the auditing process. By maintaining open lines of communication and soliciting client feedback, we ensure that our solutions are tailored to meet the unique needs and objectives of each project.

Beyond our core services, Cairo Security Clan is committed to driving innovation and shaping the future of blockchain technology. As active contributors to the ecosystem, we participate in the development of emerging technologies such as Starknet, leveraging our expertise to build robust infrastructure and tools. Through strategic guidance and support, we empower our partners to navigate the complexities of the blockchain landscape with confidence and clarity.

In summary, Cairo Security Clan stands at the forefront of blockchain security, blending technical prowess with a client-centric ethos to deliver unparalleled protection and peace of mind in an ever-evolving digital landscape. Join us in safeguarding the future of decentralized finance and digital assets with confidence and conviction.

2 Disclaimer

Disclaimer Limitations of this Audit:

This report is based solely on the materials and documentation provided by you to Cairo Security Clan for the specific purpose of conducting the security review outlined in the Summary of Audit and Scoped Files. The findings presented here may not be exhaustive and may not identify all potential vulnerabilities. Cairo Security Clan provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, occurs entirely at your own risk.

Inherent Risks of Blockchain Technology:

Blockchain technology remains in its developmental stage and is inherently susceptible to unknown risks and vulnerabilities. This review is specifically focused on the smart contract code and does not extend to the compiler layer, programming language elements beyond the reviewed code, or other potential security risks outside the code itself.

Report Purpose and Reliance:

This report should not be construed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. No third party should rely on this report for any purpose, including making investment or purchasing decisions.

Liability Disclaimer:

To the fullest extent permitted by law, Cairo Security Clan disclaims all liability associated with this report, its contents, and any related services and products arising from your use. This includes, but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services:

Cairo Security Clan does not warrant, endorse, guarantee, or assume responsibility for any products or services advertised by third parties within this report, nor for any open-source or third-party software, code, libraries, materials, or information linked to, referenced by, or accessible through this report, its content, and related services and products. This includes any hyperlinked websites, websites or applications appearing on advertisements, and Cairo Security Clan will not be responsible for monitoring any transactions between you and third-party providers. It is recommended that you exercise due diligence and caution when considering any third-party products or services, just as you would with any purchase or service through any medium.

Disclaimer of Advice:

FOR THE AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, ACCESS, AND/OR USE, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Executive Summary

This document presents the security review performed by Cairo Security Clan on the LayerAkira.

LayerAkira is a hybrid order book DEX built on the Starknet ecosystem, supporting spot asset markets. Its mission is to improve liquidity and price discovery on the Starknet ecosystem and Vision is to be a central hub where DEXs, wallets, and aggregators can access our order book liquidity.

LayerAkira team brings a rich history of building low-latency, high-frequency systems and has the backing of a large algorithmic trading firm, ensuring a seamless integration of TradFi principles in DeFi. Learn more from docs.

The audit was performed using

- manual analysis of the codebase,
- automated analysis tools,
- simulation of the smart contract,
- analysis of edge test cases

3 points of attention, where 0 is classified as Critical, 0 is classified as High, 0 is classified as Medium,1 is classified as Low,1 is classified as Informational and 1 is classified as Best Practices. The issues are summarized in Fig. 1.

This document is organized as follows. Section 1 About Cairo Security Clan. Section 2 Disclaimer. Section 3 Executive Summary. Section 4 Summary of Audit. Section 5 Risk Classification. Section 6 Issues by Severity Levels. Section 7 Test Evaluation.

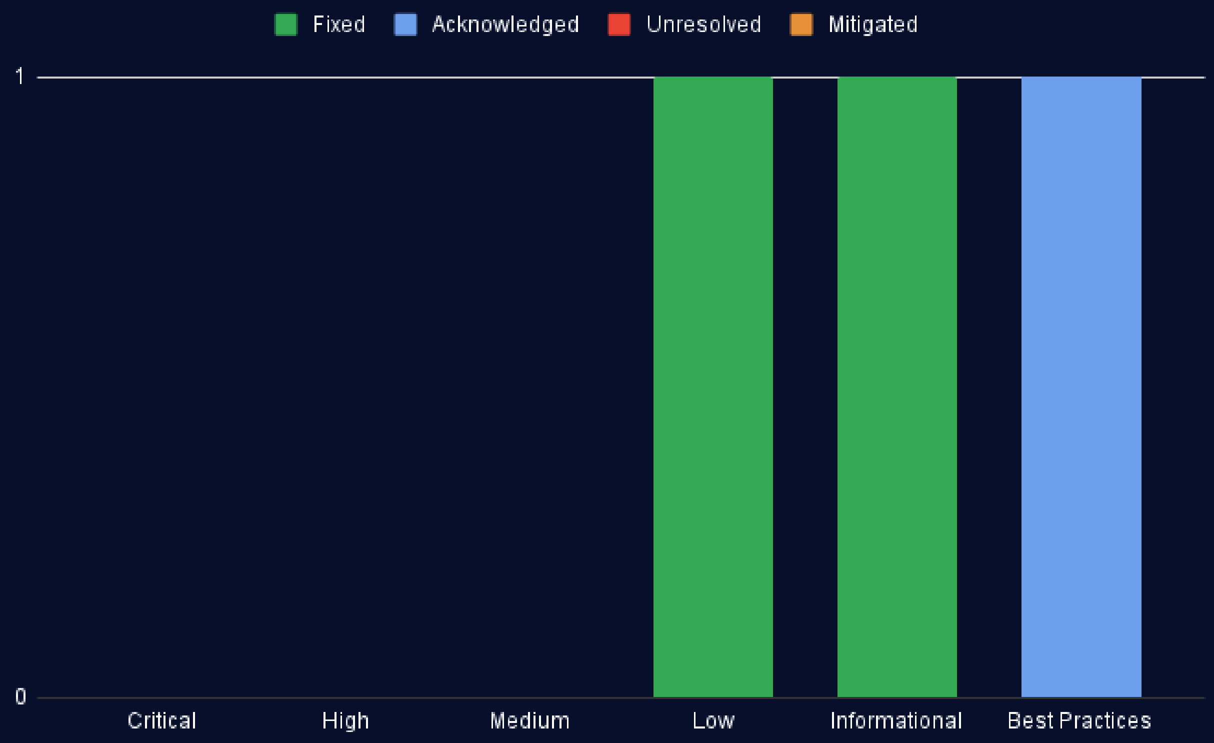

Fig 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (1), **Informational** (1), **Best Practices** (1). **Distribution of status: Fixed** (2), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0).

4 Overview

LayerAkira is a non-custodial hybrid order book DEX built on the Starknet Network. It leverages an off-chain order book and matching engine for order creation and cancellations, enabling fast trade execution unbounded by Starknet's throughput constraints. User funds are securely held in smart contracts, with trade execution and correctness ensured by on-chain settlement via rollups. The protocol combines on-chain and off-chain components. The off-chain system manages the order book and order matching, while the on-chain system handles trade settlement, fund transfers, and recording order data on the Blockchain.

Our review primarily focused on the security aspects of the on-chain component, which consists of these key components/contracts:

- `BaseTradeComponent`,
- `SORLayerAkiraExecutor`,
- `DepositComponent`,
- `WithdrawComponent`,
- `LayerAkiraCore`,
- `AccessorComponent`,
- `SignerComponent`,

and other dependent components.

- `BaseTradeComponent`: Core component that processes and settles trades by executing order matching, handling asset transfers between users, applying fees, and managing both ecosystem trades and external trades.
- `SORLayerAkiraExecutor`: Smart order router module that enables complex multi-hop trading routes by managing order paths. Designed to handle both atomic taker orders and multi-order routing scenarios, enabling efficient and secure order fulfillment.
- `DepositComponent`: Manages user deposits into the protocol by transferring tokens from users to the contract and minting corresponding internal balance tokens.
- `WithdrawComponent`: Provides functionality for securely handling token withdrawals. Supports both on-chain and off-chain withdrawals with robust validation, delay enforcement, and gas fee management.
- `LayerAkiraCore`: Central contract that orchestrates the entire protocol by integrating various components, including balance management, signer logic, deposit handling, withdrawal management, nonce tracking, and access control.
- `AccessorComponent`: Implements access control with an epoch-based system, allowing ownership management, executor whitelisting by owners, user-granted permissions to approved executors, and global permission invalidation mechanism.
- `SignerComponent`: Provides functionality for managing signer bindings and validating signatures for traders. It supports multiple signing schemes, expiration-based approvals, and integration with external signature verifiers.

5 Summary of Audit

Audit Type	Security Review
Cairo Version	2.8.5
Final Report	16/05/2025
Repository	LayerAkira/kurosawa_akira
Initial Commit Hash	d181088e434b5b5b57cfc8d9ac6b04cf59b94b80
Final Commit Hash	86165185ff4ae687facb2a710d0fb3f0e9ad31da
Documentation	Website documentation
Test Suite Assessment	High

5.1 Scoped Files

	Contracts
1	src/lib.cairo
2	src/AccessorComponent.cairo
3	src/BaseTradeComponent.cairo
4	src/DepositComponent.cairo
5	src/ExchangeBalanceComponent.cairo
6	src/Fees.cairo
7	src/LayerAkiraBaseExecutor.cairo
8	src/LayerAkiraCore.cairo
9	src/LayerAkiraExecutor.cairo
10	src/LayerAkiraExternalGrantor.cairo
11	src/NonceComponent.cairo
12	src/Order.cairo
13	src/WithdrawComponent.cairo
14	src/SORLayerAkiraExecutor.cairo
15	src/SignerComponent.cairo
16	src/signature.cairo
17	src/test_utils.cairo
18	src/testing.cairo
19	src/utils.cairo
20	src/signature/AkiraV0OffchainMessage.cairo
21	src/signature/IOffchainMessage.cairo
22	src/signature/V0OffChainMessage.cairo
23	src/utils/SlowModeLogic.cairo
24	src/utils/account.cairo
25	src/utils/common.cairo
26	src/utils/erc20.cairo

5.2 Issues

	Findings	**Severity**	**Update**
1	Certain ECDSA users unable to log in due to pubkey x-coordinate range mismatch	Low	Fixed
2	Missing Upper Bound Check for ECDSA Signature Components	Informational	Fixed
3	Missing `owner` validation in `set_owner(...)`	Best Practices	Acknowledged

6 Risk Classification

The risk rating methodology used by Cairo Security Clan follows the principles established by the CVSS risk rating methodology. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely an attacker will uncover and exploit the finding. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to Motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Likelihood		
		High	**Medium**	**Low**
Impact	**High**	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Info/Best Practices

To address issues that do not fit a High/Medium/Low severity, Cairo Security Clan also uses three more finding severities: **Informational**, **Best Practices** and **Gas**

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to formally pass to the client;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Gas** findings are used when some piece of code uses more gas than it should be or have some functions that can be removed to save gas.

7 Issues by Severity Levels

7.1 Low

7.1.1 Certain ECDSA users unable to log in due to pubkey x-coordinate range mismatch

File(s): src/SignerComponent.cairo

Description: The LayerAkira system enables traders to bind their trading accounts to a signer responsible for signing messages (such as orders), on their behalf. This binding is performed on-chain by invoking the `bind_to_signer()` function, which updates the `trader_to_signer` storage variable. This variable is a mapping of type `ContractAddress` to `ContractAddress`.

```
#[storage]
struct Storage {
    // ...
    trader_to_signer: starknet::storage::Map::<ContractAddress, ContractAddress>,
    // ...
}
```

During signature verification in the `check_sign()` function, when the signature scheme is set to ecdsa curve, the system relies on the Starknet core lib ecdsa `check_ecdsa_signature()` function. This function expects a `public_key` parameter of type `felt252`, representing the (x)-coordinate of the signer's public key point on the STARK curve.

The `felt252` type has a range of `[0, P)`, where `P = 2^251 + 17x2^192 + 1`, while the `ContractAddress` type is constrained to `[0, 2^251)`. Since the ECDSA public key is the (x)-coordinate of a point on the STARK curve, valid (x)-coordinates can also lie in the range `[2^251, P)`, which exceeds the `ContractAddress` range. Consequently, traders with public keys in this range cannot bind their signers on-chain when using the ecdsa curve signature scheme, as the `trader_to_signer` storage variable cannot store these values as `ContractAddress`. Users with valid ECDSA public keys whose (x)-coordinates fall in `[2^251, P)` will encounter binding failures, preventing them from participating in the system under the scheme ecdsa curve.

Recommendation(s): Consider using type `felt252` or enum for proper representation of signer.

Status: Fixed

Update from the client: Fixed in commit 73bb4916.

7.2 Informational

7.2.1 Missing Upper Bound Check for ECDSA Signature Components

File(s): src/SignerComponent.cairo, src/RouterComponent.cairo

Description: The `check_sign()` function of `SignerComponent` verifies signatures based on the provided signing scheme. However, for the ecdsa curve scheme, it does not verify that the signature's `r` and `s` components are less than the Stark curve order before calling `check_ecdsa_signature()`. Without these bounds checks, an out-of-range `r` or `s` could be accepted, potentially leading to unexpected behavior.

Recommendation(s): Consider checking that both `r` and `s` are below the Stark curve order before calling `check_ecdsa_signature()`.

Status: Fixed

Update from the client: Fixed in commits 1d44e557 and 86165185.

7.3 Best Practices

7.3.1 Missing `owner` validation in `set_owner(...)`

File(s): src/AccessorComponent.cairo

Description: The function `set_owner()` updates the `owner` but lacks validation to check if it's a real address. If the owner is mistakenly set to the zero address, the contract has no mechanism to recover from this state.

```
fn set_owner(ref self: ComponentState<TContractState>, new_owner: ContractAddress) {
    // @audit >> Missing zero check for new_owner
    self.only_owner();
    self.owner.write(new_owner);
    self.emit(OwnerChanged { new_owner });
}
```

Recommendation(s): Add a validation check in `set_owner()` to ensure that the `new_owner` provided is not zero address.

Status: Acknowledged

Update from the client: Our take was to eventually set owner to zero address to revoke owner access from LayerAkira.

8 Test Evaluation

8.1 Compilation Output

```
scarb build
   Compiling kurosawa_akira v0.1.0 (/contracts/Scarb.toml)
    Finished dev profile target(s) in 10 seconds
```

8.2 Tests Output

```
Collected 38 test(s) from kurosawa_akira package
Running 38 test(s) from src/
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_eth_deposit (gas: ~2481)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_indirect (gas: ~2751)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_BUY_maker_01_match_quote_qty (gas: ~4135)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_SELL_maker_03_match_base_qty (gas: ~4459)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_BUY_maker_01 (gas: ~4522)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_BUY_maker_02_match_quote_qty (gas: ~4586)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_BUY_maker_03_match_base_qty (gas: ~4522)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_SELL_maker_02_match_quote_qty (gas: ~4522)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_02::test_quote_qty_only_base (gas: ~4421)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_01::test_roter_trade_double_qty_semantic_SELL_maker_01 (gas: ~4458)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_02::test_quote_qty_both_02 (gas: ~4421)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_02::test_quote_qty_only_base (gas: ~4895)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_02::test_quote_qty_both (gas: ~4896)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_direct_no_delayed_by_exchange (gas: ~3352)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_direct_delayed (gas: ~3177)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_02::test_quote_qty_both_02 (gas: ~4896)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_SELL_maker_01 (gas: ~4004)
[PASS] kurosawa_akira::testing::tests_ecosystem_trade::test_succ_match_single_buy_taker_trade_full (gas: ~4012)
[PASS] kurosawa_akira::testing::tests_ecosystem_trade::test_succ_match_single_buy_taker_trade_full_router (gas: ~3947)
[PASS] kurosawa_akira::testing::tests_router_trade::test_execute_with_buy_taker_succ (gas: ~4393)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_SELL_maker_03_match_base_qty (gas: ~4004)
[PASS] kurosawa_akira::testing::tests_ecosystem_trade::test_succ_match_single_sell_taker_trade_full (gas: ~4067)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_SELL_maker_05_double (gas: ~4133)
[PASS] kurosawa_akira::testing::tests_router_trade::test_execute_with_buy_taker_succ_spent_side_fee_for_both_parties (gas: ~4322)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_increase_nonce (gas: ~2660)
[PASS] kurosawa_akira::testing::tests_ecosystem_trade::test_double_qty_SELL_maker_01_oracle_qty (gas: ~4068)
[PASS] kurosawa_akira::testing::tests_router_trade::test_cant_execute_with_not_registered_router (gas: ~2792)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_SELL_maker_04_double (gas: ~4133)
[PASS] kurosawa_akira::testing::tests_router_trade::test_execute_with_sell_taker_succ (gas: ~4452)
[PASS] kurosawa_akira::testing::tests_ecosystem_trade::test_succ_match_single_sell_taker_trade_full_router (gas: ~4001)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_direct_delayed_cant_apply_twice (gas: ~3209)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_indirect_twice (gas: ~2788)
[PASS] kurosawa_akira::testing::tests_router_trade::test_punish_router (gas: ~3654)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_01::test_double_qty_SELL_maker_02_match_quote_qty (gas: ~4132)
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_02::test_quote_qty_both (gas: ~4421)
[PASS] kurosawa_akira::testing::tests_deposit_and_withdrawal_and_nonce::test_withdraw_eth_direct_immediate (gas: ~3213)
```

```
[PASS] kurosawa_akira::testing::tests_quote_qty_ecosystem_trade_02::test_quote_qty_only_quote (gas: ~4422)
[PASS] kurosawa_akira::testing::tests_quote_qty_router_trade_02::test_quote_qty_only_quote (gas: ~4896)
Tests: 38 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
```