CAIRO SECURITY CLAN

STARKNET PAYMENTS

SECURITY ASSESSMENT REPORT

SEPTEMBER 2025

Prepared for

STARKWARE

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 Scope. 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 Starknet Payments developed by Starkware.

Starknet Payments provides an on-chain order matching contract where trades are executed after off-chain orders are matched and validated through signatures. The system, enables secure and efficient decentralized payment and trading flows on Starknet.

The audit was performed using

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

There are 3 points of attention, where 1 are classified as Critical, 0 as High, 0 as Medium, 0 as Low, 1 as Informational, 1 as Best Practices, and 0 as Undetermined. 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.

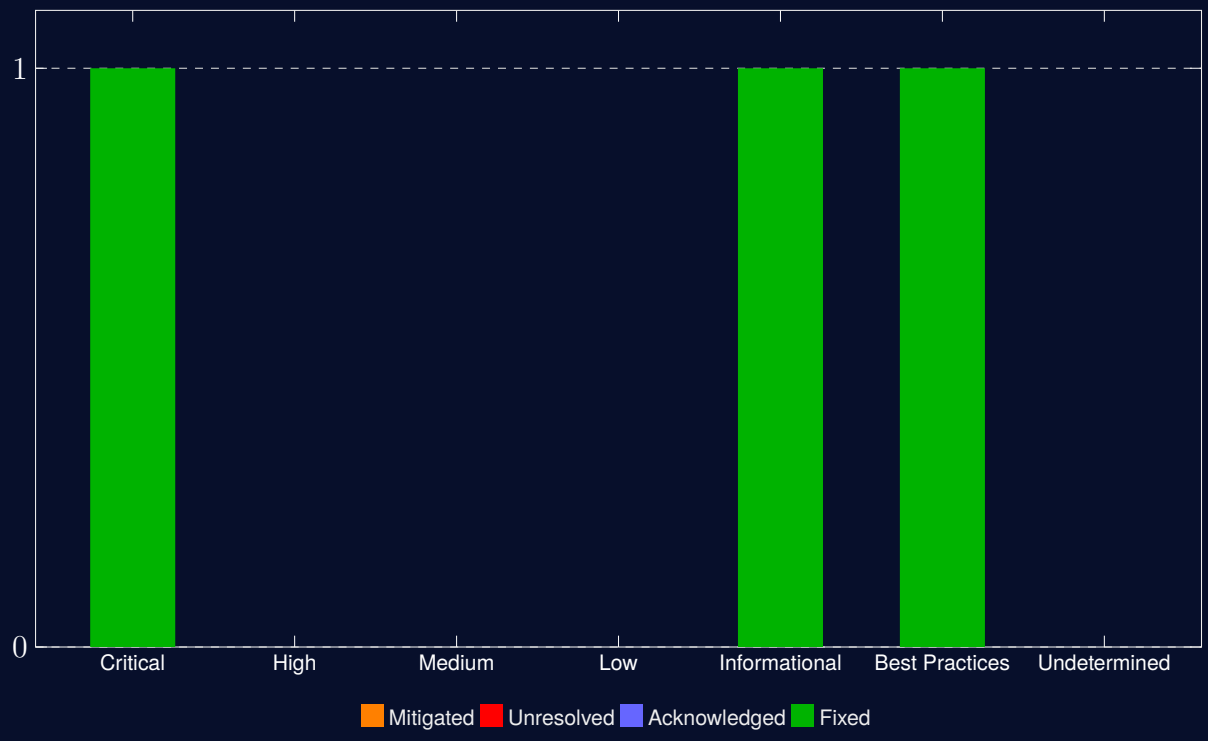

Fig 1: Distribution of issues: Critical (1), **High** (0), **Medium** (0), **Low** (0), **Informational** (1), **Best Practices** (1), **Undetermined** (0). **Distribution of status: Fixed** (3), **Acknowledged** (0), **Mitigated** (0), **Unresolved** (0).

4 Summary of Audit

Audit Type	Security Review
Cairo Version	2.11.4
Final Report	29/09/2025
Repository	starkware-libs/starknet-payments
Initial Commit Hash	03e917bf5d722b0c1e54d0ba7def007dc42152c7
Test Suite Assessment	High

4.1 Scoped Files

	Contracts
1	/src/lib.cairo
2	/src/order.cairo
3	/src/payments.cairo
4	/src/tests.cairo
5	/src/utils.cairo
6	/src/errors.cairo
7	/src/events.cairo
8	/src/interface.cairo

4.2 Issues

	Findings	Severity	Status
1	Dust fills drain both orders into protocol fees because price checks ignore net amounts	Critical	Fixed
2	Event data omits fees and net amounts	Informational	Fixed
3	Avoid zero-value ERC20 transfers after fees	Best Practices	Fixed

5 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.

6 Issues by Severity Levels

6.1 Critical

6.1.1 Dust fills drain both orders into protocol fees because price checks ignore net amounts

File(s): /src/payments.cairo

Description: `_validate_settlement` enforces each order's limit price using the matcher-supplied gross amounts before fees. `trade` then debits both users' balances by a protocol fee computed with ceiling division before it transfers the net settlement amounts using `_calculate_fee`. For any positive fee, a dust fill with `order_a_actual_sell_amount = order_a_actual_buy_amount = 1` satisfies the ratio constraints for two symmetric 1:1 orders, but both `ceil(amount * fee / 10_000)` evaluations return 1. As a result the net transfers become zero while each trader still pays one token of fees, and the fulfillment counter advances. An arbitrary relayer can repeat this on public orders until both sides have lost their entire positions to the fee recipient, despite never receiving the agreed assets. More generally, by choosing minimal actual amounts the relayer can make the post-fee price arbitrarily worse than the signed price, completely bypassing user slippage limits. This breaks the core settlement guarantee and enables deterministic loss of user funds whenever `fee > 0`.

Recommendation(s): Consider checking minimum amounts and ensure the amounts received correctly.

Status: Fixed

Update from the client: Fixed at PR #23

6.2 Informational

6.2.1 Event data omits fees and net amounts

File(s): /src/events.cairo

Description: `TradeExecuted` only includes the gross `order_a_sell_amount` and `order_a_buy_amount`. This makes it difficult for indexers and users to reconstruct net settlement after fees and to verify execution quality on-chain.

```
#[derive(Debug, Drop, PartialEq, starknet::Event)]
pub struct TradeExecuted {
    #[key]
    pub user_a: ContractAddress,
    #[key]
    pub user_b: ContractAddress,
    #[key]
    pub sell_token: ContractAddress,
    #[key]
    pub buy_token: ContractAddress,
    pub order_a_sell_amount: u128,
    pub order_a_buy_amount: u128,
}
```

Recommendation(s): Extend `TradeExecuted` with `fee_a`, `fee_b`, and net amounts or emit a separate `FeesCollected` event.

Status: Fixed

Update from the client: Fixed at 621bee3

6.3 Best Practices

6.3.1 Avoid zero-value ERC20 transfers after fees

File(s): /src/payments.cairo

Description: After fee deduction, the code may attempt `transfer_from(..., 0)` when `fee >= amount`. While many ERC20 implementations allow zero-amount transfers, some non-standard tokens revert. Given tokens are explicitly whitelisted, this is unlikely but avoidable.

Recommendation(s): Short-circuit transfers when the post-fee net is zero to avoid unnecessary external calls and potential incompatibilities.

Status: Fixed

Update from the client: Fixed at d8bbe8e

7 Compilation & Test Output

7.1 Compilation Output

```
scarb build
    Updating git repository https://github.com/starkware-libs/starkware-starknet-utils
 Downloading openzeppelin_introspection v2.0.0
 Downloading openzeppelin_finance v2.0.0
 Downloading openzeppelin_presets v2.0.0
 Downloading openzeppelin_upgrades v2.0.0
 Downloading snforge_scarb_plugin v0.45.0
 Downloading openzeppelin v2.0.0
 Downloading openzeppelin_merkle_tree v2.0.0
 Downloading openzeppelin_access v2.0.0
 Downloading openzeppelin_token v2.0.0
 Downloading snforge_std v0.45.0
 Downloading openzeppelin_governance v2.0.0
 Downloading openzeppelin_security v2.0.0
 Downloading openzeppelin_account v2.0.0
 Downloading openzeppelin_testing v4.2.0
 Downloading openzeppelin_utils v2.0.0
   Compiling starknet_payments v0.1.0 (/contracts/Scarb.toml)
    Finished `dev` profile target(s) in 53 seconds
```

7.2 Tests Output

```
scarb test
     Running test starknet_payments (SNFORGE_BACKTRACE=1 snforge test)
Warning:  To get accurate backtrace results, it is required to use the configuration available in the latest
    Cairo version. For more details, please visit https://foundry-rs.github.io/starknet-foundry/snforge-advanced-
    features/backtrace.html
   Compiling test(starknet_payments_unittest) starknet_payments v0.1.0 (contracts/Scarb.toml)
    Finished `dev` profile target(s) in 40 seconds


Collected 9 test(s) from starknet_payments package
Running 9 test(s) from src/
[PASS] starknet_payments::order::tests::test_order_type_hash (l1_gas: ~0, l1_data_gas: ~0, l2_gas: ~6760000)
[PASS] starknet_payments::tests::test_payments::test_failed_address_allowlist (l1_gas: ~0, l1_data_gas: ~2304,
    l2_gas: ~8565120)
[PASS] starknet_payments::tests::test_payments::test_failed_handle_order (l1_gas: ~0, l1_data_gas: ~2208, l2_gas:
     ~7544640)
[PASS] starknet_payments::tests::test_payments::test_failed_register_token (l1_gas: ~0, l1_data_gas: ~2304,
    l2_gas: ~8565120)
[PASS] starknet_payments::tests::test_payments::test_failed_set_fee (l1_gas: ~0, l1_data_gas: ~2208, l2_gas:
    ~8064640)
[PASS] starknet_payments::tests::test_payments::test_successful_address_allowlist (l1_gas: ~0, l1_data_gas:
    ~2208, l2_gas: ~10746560)
[PASS] starknet_payments::tests::test_payments::test_successful_handle_order (l1_gas: ~0, l1_data_gas: ~2496,
    l2_gas: ~11086080)
[PASS] starknet_payments::tests::test_payments::test_successful_register_token (l1_gas: ~0, l1_data_gas: ~2208,
    l2_gas: ~10746560)
[PASS] starknet_payments::tests::test_payments::test_successful_set_fee (l1_gas: ~0, l1_data_gas: ~2208, l2_gas:
    ~8950720)
Tests: 9 passed, 0 failed, 0 ignored, 0 filtered out
```