



SMART CONTRACT SECURITY ASSESSMENT

SparkFi

V12 Contracts

CLIENT

SparkFi

REVIEW

July 2024

DOCUMENT

Document information

Project	SparkFi, V12 Contracts
Client	SparkFi
Platform	Not recorded in the original report
Language	Not recorded in the original report
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Automated scanning, manual code review
Date of engagement	15 to 22 July 2024

Document history

VERSION	DATE	CHANGE
1.0	July 2024	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	3
02	Scope	4
03	Methodology	5
04	Severity and status	6
05	Findings summary	7
06	Detailed findings	8
	H-01 Missing Persistent Update in increase_user_volume	8
	H-02 Front-Running Vulnerability in Matching Functions	8
	M-01 Missing Restriction for IOC Orders	9
	M-02 Lack of Vector Length Restriction	9
	L-01 Missing Documentation	9
	L-02 log_order_change_info Emits No Logs	10
	L-03 Dead Code in mul_div_rounding_up	10
	L-04 Ineffective require in order_id	10
07	Conclusion	11
	Disclaimer	12

Executive summary

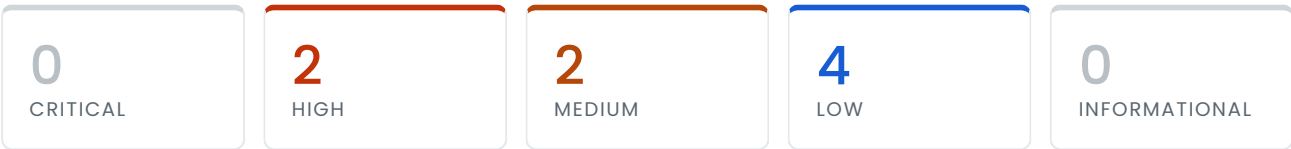
About SparkFi

SparkFi V12 is an order book exchange with market, limit and immediate-or-cancel (IOC) order matching and per-user trading volume tracking.

Summary of the review

Safe Edges audited SparkFi V12 between 15 and 22 July 2024 using automated scanning and manual code review. Eight issues were identified: two High, two Medium and four Low. The High issues were user trading volume that was updated in memory but never written to storage, and public matching functions that can be front-run. All eight were unresolved at the time of the report.

Findings by severity



Findings by status

STATUS	ISSUES
● Open	8

Scope

Repository	Not recorded in the original report
Commit	Not recorded in the original report
Platform	Not recorded in the original report
Language	Not recorded in the original report

Only the code at the commit above was reviewed. Changes made after this commit are not covered by this report.

Methodology

Methods recorded for this engagement: Automated scanning, manual code review.

Manual review

Auditors read the in-scope code line by line, tracing control flow, state changes and trust boundaries against the intended behaviour described by the client.

Threat modelling

We map privileged roles, external dependencies and value flows, then test the assumptions the code makes about each of them.

Functional and automated testing

Where tooling exists for the language and chain, we run static analysis and targeted tests to confirm suspected issues and check edge cases.

Fix review

After the client addresses the findings, we review the updated code and record the final status of each issue.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
H-01	Missing Persistent Update in increase_user_volume	High	● Open
H-02	Front-Running Vulnerability in Matching Functions	High	● Open
M-01	Missing Restriction for IOC Orders	Medium	● Open
M-02	Lack of Vector Length Restriction	Medium	● Open
L-01	Missing Documentation	Low	● Open
L-02	log_order_change_info Emits No Logs	Low	● Open
L-03	Dead Code in mul_div_rounding_up	Low	● Open
L-04	Ineffective require in order_id	Low	● Open

Detailed findings

H-01 Missing Persistent Update in increase_user_volume

Severity **High** Status ● Open

DESCRIPTION

In `increase_user_volume`, the logic performs an in-memory update but fails to persist the updated state in storage. The expected behavior is to persist changes after calculating and updating the user's volume.

IMPACT

The lack of persistence means user trading volume tracking will be inaccurate.

RECOMMENDATION

Ensure that the updated `user_volume` is written back to storage.

H-02 Front-Running Vulnerability in Matching Functions

Severity **High** Status ● Open

DESCRIPTION

Functions like `match_market_order` and `match_ioc_order` are public and can be front-run. An attacker can predictably exploit these by observing the mempool and preempting orders.

IMPACT

Results in market manipulation or preferential order execution.

RECOMMENDATION

Add mechanisms to protect against front-running, such as commit-reveal schemes or MEV-resistant designs.

M-01 Missing Restriction for IOC Orders

Severity **Medium** Status ● Open

DESCRIPTION

The `match_ioc_order` function can be called with limit orders too, which breaks the intent of IOC behavior.

IMPACT

Incorrect order execution logic.

RECOMMENDATION

Add type-checking to ensure only IOC orders are passed.

M-02 Lack of Vector Length Restriction

Severity **Medium** Status ● Open

DESCRIPTION

Several functions take in `Vec<T>` inputs but do not restrict their maximum length.

IMPACT

Potential denial-of-service from large inputs or excessive gas use.

RECOMMENDATION

Set and enforce maximum length for vectors in function parameters.

L-01 Missing Documentation

Severity **Low** Status ● Open

DESCRIPTION

Several functions lack clear documentation or NatSpec comments.

IMPACT

Poor developer experience and risk of misunderstanding the intended behavior.

RECOMMENDATION

Add descriptive comments and usage examples.

L-02 `log_order_change_info` Emits No Logs

Severity Low Status ● Open

DESCRIPTION

The `log_order_change_info` function is defined and called but emits no log.

IMPACT

Pointless function execution and wasted gas.

RECOMMENDATION

Either remove the function or ensure it emits meaningful logs.

L-03 `Dead Code in mul_div_rounding_up`

Severity Low Status ● Open

DESCRIPTION

The `mul_div_rounding_up` function is never used.

IMPACT

Increased contract size and audit overhead.

RECOMMENDATION

Remove unused code.

L-04 `Ineffective require in order_id`

Severity Low Status ● Open

DESCRIPTION

The require condition `require(token_id_a != token_id_b)` in `order_id` may be trivial since validation is already assumed elsewhere.

IMPACT

Code redundancy.

RECOMMENDATION

Evaluate necessity and remove if truly redundant.

Conclusion

The SparkFi V12 contracts are well-structured but exhibit key security oversights involving storage persistence, front-running vectors, missing validations, and documentation gaps. Prompt implementation of the above recommendations is essential to enhance the protocol's security posture.

Safe Edges remains available for a follow-up audit to verify remediations.

Disclaimer

This report is based on the code at the commit listed in the Scope section and on information the client provided during the review. It reflects Safe Edges' assessment at the time of the review and is not a guarantee that the code is free of vulnerabilities.

Changes made after the reviewed commit, deployment configuration, off-chain components and third-party dependencies are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS