RISK AUDIT

for

CREDIT COOP

on

February 12, 2025

FIDESIUM

Executive Summary

Report

32
/100

TOTAL
Low risk
February 12, 2025

Abstract

Fidesium's automated risk assessment service was requested to perform a risk posture audit on CreditCOOP **contracts**

Repository Link: https://github.com/credit-cooperative/Line-Of-Credit

Initial Commit Hash:

```
a9c13c109a5a5389639cd9508cc7637720fcab05
```

Issue Summary

🟥 Critical 2 Issues	🟧 High 3 Issues	🟨 Medium 6 Issues	🟩 Low 2 Issues	⬜ Info 2 Issues

Caveats

FourBy's codebase is well written, but does incur a handful of low risk flaws.

Test Approach

Fidesium performed both Whitebox and Blackbox testing, as per the scope of the engagement, and relied on automated security testing.

Methodology

The assessment methodology covered a range of phases and employed various tools, including but not limited to the following:

- Mapping Content and Functionality of API
- Application Logic Flaws
- Access Handling
- Authentication/Authorization Flaws
- Brute Force Attempt
- Input Handling
- Source Code Review
- Fuzzing of all input parameter
- Dependency Analysis

Severity Definitions

Critical	The issue can cause large economic losses, large-scale data disorder or loss of control of authority management.
High	The issue puts users' sensitive information at risk or is likely to lead to catastrophic financial implications.
Medium	The issue puts a subset of users' sensitive information at risk, reputation damage or moderate financial impact.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is low-impact to the client's business.
Informational	The issue does not pose an immediate risk but is relevant to security best practices or defence in Depth.

Risk Issues

Vunerability	Description	Risk	Probability	Status
Oracle Manipulation/Flash Loan	Multiple contracts are vulnerable to oracle manipulation.	Critical	Low	Active
Arbitrary Calldata passing	The `SpigotLib` contract passes arbitrary calldata	Critical	Low	Active
Missing Sequencer Uptime	Multiple L2 Oracles do not conduct a sequencer uptime check	High	Medium	Active
Unprotected One Time Setup function	The `LineOfCredit` contract lacks `init` protection	High	Medium	Active
One Step Ownership Transfer	Multipe Contracts Implement a one step ownership transfer	High	Medium	Active
Centralization	Contracts have Priviliged Roles	Medium	Medium	Active
Long Price Feed Latency	Oracle Contracts use a single latency of `25 hours`	Medium	Medium	Active
Missing Zero Address Validation	Multiple locations in the codebase are missing a zero address validation. This can result in unexpected behavior, and lost assets.	Medium	Medium	Active
Missing Contract Address Validation	Multiple locations in the codebase are missing a contract address validation. This can result in unexpected behavior, and lost assets.	Medium	Medium	Active
Missing Pausability	The contract do not allow pausing. This could limit the ability of the developer to respond in an emergency.	Medium	Medium	Active
Reliance on Block Timestamp	Multiple functions rely on `block.timestamp`.	Medium	Unlikely	Active
Missing bound validations	Multiple parameters lack upper/lower bound validations. This could result in excessively high fees and other issues.	Low	Low	Active
Missing zero bytes validation	Multiple locations in the codebase are zero bytes validations. This could lead to accounting erros, or functionality bypassing	Low	Low	Active
Gas Optimization: Unnecessary `uint256`	The contract implements `uint256` for multiple variables and parameters.	Info	Info	Active
Gas Optimization: Unnecessary storage reads	The `SpigotedLine` contract executes unnecessary storage reads.	Info	Info	Active

Risk Overview

Team Risk

Low risk: 1

No issues found in founding team

Doxxing Status	Team Experience	Risk Summary
Public	Highly relevant	Low

Smart Contract Risks

Risk summary: 36

The contracts are mostly well written, but have a handful of flaws that should to be carefuly managed.

Vulnerabilities Critical

Oracle Manipulation/Flash Loan

Vulnerability severity: **Critical**

Vulnerability probability: **Low**

Multiple contracts are vulnerable to oracle manipulation and are using a single oracle/pricefeed.

An attacker could either trigger a flashloan, or monitor oracle update frequency and time transactions to hit price boundaries

- LineOfCredit
- Escrow
- LineFactory
- ArbitrumOracle
- BaseOracle
- Oracle
- zkEVMOracle
- CreditLib

Recommendations:

- Implement a TWAP Oracle with manipulation checks
- Implement constant circuit breakers for max daily usage per wallet
- Implement multi oracle price feeds
- Implement oracle freshness checks

Arbitrary Calldata passing

Vulnerability severity: **Critical**

Vulnerability probability: **Low**

The `SpigotLib` contract passes arbitrary calldata

An attacker could construct malicious contracts or pass malicious data to self destruct, manipulate state, or have other unexpected effects

```
function _claimRevenue(
    ...
(bool claimSuccess, ) = revenueContract.call(data);
...
```

Recommendations:

- Ensure interface compliance, the function selector, and parameters match expectation. Use `abi.decode` to identify params, `bytes4` to identify the selector.
- Alternatively whitelist known good addresses

Vulnerabilities High

Missing Sequencer Uptime

Vulnerability severity: **High**

Vulnerability probability: **Medium**

Multiple L2 Oracles do not conduct a sequencer uptime check

This could lead to economic exploits and loss of funds.

- ArbitrumOracle
- BaseOracle
- zkEVMOracle

Recommendations:

Implement a sequencer uptime check:

```
function isSequencerActive() internal view returns (bool) {
                        ArbSys arbSys = ArbSys(address(100));
                        uint256 lastBlockTime = block.timestamp - block.number + arbSys.arbBlockNumber();
                        return block.timestamp - lastBlockTime < MAX_PRICE_LATENCY;
                    }
```

Unprotected One Time Setup function

Vulnerability severity: **High**

Vulnerability probability: **Medium**

The `LineOfCredit` contract lacks `init` protection

This function is external and reverts after initial execution. An attacker could frontrun execution and call this function at an unexpeted time, or with unexpected state.

Recommendations:

There are a handful of options, sorted in order of security

- Apply the constructor time initialization pattern
- Set `msg.sender` to a variable in `constructor` and validate `init` uses the same `msg.sender`

Vulnerabilities High

One Step Ownership Transfer

Vulnerability severity: **High**

Vulnerability probability: **Medium**

Multipe Contracts Implement a one step ownership transfer

This could lead to loss of contract control.

- ArbitrumOracle
- BaseOracle
- PolygonOracle
- zkEVMOracle
- SpigotLib

Recommendations:

Implement a two step ownership transfer

Vulnerabilities Medium

Centralization

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Contracts have Priviliged Roles

Contract	Role
SpigotedLine	arbiter
LineFactory	arbiter
ArbitrumOracle	owner
BaseOracle	owner
Oracle	owner
zkEVMOracle	owner
SpigotLib	self.owner
SpigotLib	self.operator

Recommendations:

Please ensure priviliged roles are well managed multisigs.

Long Price Feed Latency

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Oracle Contracts use a single latency of `25 hours`

This is a long period for volatile assets

- ArbitrumOracle
- BaseOracle
- Oracle
- PolygonOracle
- zkEVMOracle

Recommendations:

Reduce latency, and introduce per asset heartbeats

Vulnerabilities Medium

Missing Zero Address Validation

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Multiple locations in the codebase are missing a zero address validation. This can result in unexpected behavior, and lost assets.

Contract	Function	Parameter
EscrowedLine	constructor	_escrow
EscrowedLine	_liquidate	to
EscrowedLine	_liquidate	targetToken
EscrowedLine	_rollover	newLine
SecuredLine	constructor	oracle_
SecuredLine	constructor	arbiter_
SecuredLine	constructor	borrower_
SecuredLine	constructor	swapTarget_
SecuredLine	constructor	spigot_
SecuredLine	constructor	escrow_
SecuredLine	rollover	newLine
SecuredLine	liquidate	targetToken
SpigotedLine	constructor	oracle_
SpigotedLine	constructor	arbiter_
SpigotedLine	constructor	borrower_
SpigotedLine	constructor	spigot_
SpigotedLine	constructor	swapTarget_
SpigotedLine	claimAndReplay	claimToken
SpigotedLine	claimAndTrade	claimToken
SpigotedLine	_claimAndTrade	claimToken
SpigotedLine	_claimAndTrade	targetToken
SpigotedLine	updateOwnerSplit	revenueContract
SpigotedLine	addSpigot	revenueContract
SpigotedLine	releaseSpigot	to
SpigotedLine	sweep	to

continued ...

Vulnerabilities Medium

Missing Zero Address Validation - continued

Contract	Function	Parameter
SpigotedLine	sweep	token
SpigotedLine	tradeable	token
SpigotedLine	unused	token
Escrow	constructor	_oracle
Escrow	constructor	_line
Escrow	constructor	_borrower
Escrow	updateLine	_line
Escrow	addCollateral	token
Escrow	enableCollateral	token
Escrow	releaseCollateral	token
Escrow	releaseCollateral	to
Escrow	liquidate	token
Escrow	liquidate	to
LineFactory	constructor	moduleFactory
LineFactory	deployEscrow	owner
LineFactory	deployEscrow	borrower
LineFactory	deploySpitgot	owner
LineFactory	deploySpitgot	operator
LineFactory	registerSecuredLine	line
LineFactory	registerSecuredLine	spigot
LineFactory	registerSecuredLine	escrow
LineFactory	registerSecuredLine	borrower
LineFactory	registerSecuredLine	operator
LineFactory	rolloverSecuredLine	oldLine
LineFactory	rolloverSecuredLine	borrower
ModuleFactory	deploySpigot	owner
ModuleFactory	deploySpigot	operator
ModuleFactory	deployEscrow	oracle

continued ...

Vulnerabilities Medium

Missing Zero Address Validation - continued

Contract	Function	Parameter
ModuleFactory	deployEscrow	owner
ModuleFactory	deployEscrow	borrower
ArbitrumOracle	_getLatestAnswer	token
ArbitrumOracle	setOwner	_owner
ArbitrumOracle	setPriceFeed	token
ArbitrumOracle	setPriceFeed	feed
BaseOracle	setPriceFeed	feed
BaseOracle	setPriceFeed	token
BaseOracle	getLatestAnswer	token
BaseOracle	_getLatestAnswer	token
BaseOracle	setOwner	token
Spigot	constructor	_owner
Spigot	constructor	_operator
Spigot	claimRevenue	revenueContract
Spigot	claimRevenue	token
Spigot	claimOwnerTokens	token
Spigot	claimOperatorTokens	token
Spigot	operate	revenueContract
Spigot	addSpigot	revenueContract
Spigot	removeSpigot	revenueContract
Spigot	updateOwnerSplit	revenueContract
Spigot	updateOwner	newOwner
Spigot	updateOperator	newOperator
Spigot	getOwnerTokens	token
Spigot	getOperatorTokens	token
Spigot	getSetting	revenueContract

Recommendations:

Use `!= address(0)` to validate these parameters are not zero addresses

Vulnerabilities Medium

Missing Contract Address Validation

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Multiple locations in the codebase are missing a contract address validation. This can result in unexpected behavior, and lost assets.

Contract	Function	Contract
EscrowedLine	constructor	_escrow
LineFactory	constructor	moduleFactory
ArbitrumOracle	setPriceFeed	feed
BaseOracle	setPriceFeed	feed
Oracle	setPriceFeed	feed
PolygonOracle	setPriceFeed	feed
zkEVMOracle	setPriceFeed	feed

Recommendations:

- Use `Address.isContract()` to validate that these are a valid contract.
- Validate code length, e.g. `targetAddress.code.length != 0`
- Validate key ERC20 abi functions, eg:

```
try SafeERC20(contractAddress).totalSupply() returns (uint256) {
    return true;
} catch {
    return false;
}
```

Missing Pausability

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Multple contracts do not allow pausing. This could limit the ability of the developer to respond in an emergency.

Recommendations:

Use `Pausable` from OpenZeppelin

Vulnerabilities Medium

Reliance on Block Timestamp

Vulnerability severity: **Medium**

Vulnerability probability: **Unlikely**

Multiple functions rely on `block.timestamp`, which can be manipulated by miners.

Contract	Function
LineOfCredit	constructor
LineOfCredit	healthcheck
InterestRateCredit	accrueInterest
InterestRateCredit	_accrueInterest
InterestRateCredit	_calculateInterestOwed
ArbitrumOracle	getLatestAnswer
ArbitrumOracle	_getLatestAnswer
BaseOracle	getLatestAnswer
BaseOracle	_getLatestAnswer
Oracle	getLatestAnswer
Oracle	_getLatestAnswer
PolygonOracle	getLatestAnswer
PolygonOracle	_getLatestAnswer
zkEVMOracle	getLatestAnswer
zkEVMOracle	_getLatestAnswer
SBCPriceFeedPolygon	latestRoundData
stUSDriceFeedArbitrum	latestRoundData

Recommendations:

- Use block numbers instead of timestamps.
- If timestamps are necessary, use trusted external oracles.

Vulnerabilities Low

Missing bound validations

Vulnerability severity: **Low**

Vulnerability probability: **Low**

Multiple parameters lack upper/lower bound validations. This could result in excessively high fees and other issues.

Contract	Function	Parameter
EscrowedLine	_liquidate	amount
SecuredLine	constructor	defaultSplit_
SecuredLine	liquidate	amount
Escrow	constructor	_minimumCollateralRatio
LineFactory	deployEscrow	minCRatio
LineFactory	deploySecuredLine	ttl
LineFactory	deploySecuredLineWithConfig	coreParams.revenueSplit
LineFactory	registerSecuredLine	revenueSplit
LineFactory	registerSecuredLine	minCRatio
LineFactory	rolloverSecuredLine	ttl

Recommendations:

Implement lower and upper bound validations

Vulnerabilities Low

Missing zero bytes validation

Vulnerability severity: **Low**

Vulnerability probability: **Low**

Multiple locations in the codebase are zero bytes validations. This could lead to accounting erros, or functionality bypassing

Contract	Function	Parameter
EscrowedLine	_liquidate	id
LineOfCredit	mutualConsentById	id
LineOfCredit	setRates	id
LineOfCredit	increaseCredit	id
LineOfCredit	close	id
LineOfCredit	borrow	id
LineOfCredit	withdraw	id
LineOfCredit	available	id

Recommendations:

Use `!= bytes32(0)` to validate all `bytes32` parameters.

Vulnerabilities Info

Gas Optimization: Unnecessary `uint256`

Vulnerability severity: **Info**

Vulnerability probability: **Info**

The contract implements `uint256` for multiple variables and parameters.

Contract	Function	Parameter
EscrowedLine	_liquidate	returns
EscrowedLine	_liquidate	amount
LineOfCredit	**MULTIPLE LOCATIONS**	**MULTIPLE LOCATIONS**
SecuredLine	liquidate	amount
SpigotedLine	**MULTIPLE LOCATIONS**	**MULTIPLE LOCATIONS**
Escrow	releaseCollateral	amount
Escrow	releaseCollateral	returns
Escrow	getCollateralRatio	returns
Escrow	getCollateralValue	returns
Escrow	liquidate	amount

This might consume unnecessary gas

Recommendations:

Validate against business logic to ensure that you can not rely on smaller numbers such as `uint64`

Gas Optimization: Unnecessary storage reads

Vulnerability severity: **Info**

Vulnerability probability: **Info**

The `SpigotedLine` contract executes unnecessary storage reads.

`credit.token` is read mutiple times in `useAndRepay`

This might consume unnecessary gas

Recommendations:

Cache `credit.token`

Disclaimer

Disclaimer

This report is governed by the Fidesium terms and conditions.

This report does not constitute an endorsement or disapproval of any project or team, nor does it reflect the economic value or potential of any related product or asset. It is not investment advice and should not be used as the basis for investment decisions. Instead, this report provides an assessment intended to improve code quality and mitigate risks inherent in cryptographic tokens and blockchain technology.

Fidesium does not guarantee the absence of bugs or vulnerabilities in the technology assessed, nor does it comment on the business practices, models, or regulatory compliance of its creators. All services, reports, and materials are provided "as is" and "as available," without warranties of any kind, including but not limited to merchantability, fitness for a particular purpose, or non-infringement.

Cryptographic assets and blockchain technologies are novel and carry inherent technical risks, uncertainties, and the possibility of unpredictable outcomes. Assessment results may contain inaccuracies or depend on third-party systems, and reliance on them is solely at the Customer’s risk.

Fidesium assumes no liability for content inaccuracies, personal injuries, property damages, or losses related to the use of its services, reports, or materials. Third-party components are provided "as is," and any warranties are strictly between the Customer and the third-party provider.

These services and materials are intended solely for the Customer's use and benefit. No third party or their representatives may claim rights to or rely on these services, reports, or materials under any circumstances.