# Security Assessment Report

# Cables Finance

## Soroban Platform

March 2024

certora

cables

# Table of content

certora

cables

# Project Summary

## Project Scope

| Repo Name | Repository | Last Commit | Compiler version | Platform |
|---|---|---|---|---|
| cables_stellar_contracts | https://github.com/ShiftForex/cables_stellar_contracts | `686a18c4f9ffd261baa39faf091732fc42e49c37` | Soroban-sdk 20.3.2 | Stellar Soroban |

## Project Overview

The purpose of this research was to audit all components of the Cables protocol, with the goal of validating the architecture and design, and strengthening any assumptions of soundness, security and robustness of the DeFi application.
We were initially supplied with a given commit, and were later on redirected to a newer branch containing some optimization and API changes - including the addition of the `match_and_fill` function.
The audit was performed from February to March 2024.

## Stellar and Soroban Overview

Stellar is a blockchain platform, on which Soroban provides a smart-contract runtime.
Soroban is powered by a WASM VM and provides a toolchain for developing and deploying contracts written in Rust.

At the time of this document's writing, Soroban is still in very early stages, and as such there exists very little material, both in terms of existing contracts and in security research.

Soroban provides a fairly simple authentication interface, allowing contract developers to validate that specific addresses have signed (fully or partially) for the current invocation.

Other environmental interfaces exist in Soroban, for on-ledger persistent and temporary storage, as well as various arithmetic and cryptographic primitives.

All in all - Soroban contracts are idiomatic from the perspective of a Rust developer, and require significantly less scrutiny with regard to unusual code and in-code configurations, compared to Solana for example.

## Cables Overview

Cables implement an order book and escrow, allowing users to place several types of parameterized orders exchanging between assets.
Upon placing an order, the user deposits the maximum amount to be paid out in the transaction, which is credited to the order balance.
Orders are filled by so-called "execution facilities" - which define specific addresses that are allowed to fill two opposite orders (buy & sell) with matching parameters.
An execution facility may also perform a "match and fill" - which simply means for a given order - create the opposite order and then fill both.

The logic driving the execution facilities, such as the priority of orders, matching algorithm, timing, and so on is located off-chain and is out of the scope of this audit.

To summarize - users interact with the contract by placing orders and their funds in escrow, while trusted execution facilities are driven off-chain to fill matching orders.

## Findings Summary

The table below summarizes the findings of the review, including type and severity details.

| Severity | Discovered | Acknowledged | Code Fixed |
|---|---|---|---|
| Critical | - | - | - |
| High | - | - | - |
| Medium | 3 | 3 | 2 |
| Low | 2 | 2 | 2 |
| Informational | 5 | 5 | 5 |
| **Total** | 10 | 10 | 9 |

certora cables

# Detailed Findings

## Medium Severity Issues

### M-1. Front-running exposure through order cancellation

Description:

Since users are able to cancel their orders without paying any fine - a user who is front-running transactions may put up broad limit orders and prevent them from being filled by submitting a cancel request before they are filled.

This may allow users to potentially manipulate the price of certain assets and take advantage of market conditions while taking no risk and ensuring that no losses are actually incurred.

Recommendation:

We recommend that canceled orders be charged a cancellation fee to offset and deter unwarranted cancellations.

Customer's response:

Acknowledged. 60 second delay for order cancellation was introduced to the contract.

Commit: 09e6db7d224a07f077294ffcc89f410c9519ee73

https://github.com/ShiftForex/cables_stellar_contracts/tree/09e6db7d224a07f077294ffcc89f410c9519ee73

Fix review: OK

### M-2. Loss of funds when trading against malicious asset

Description:

Due to how the amount to deposit is computed at the point of order creation, and the following calculations when filling that order - a malicious asset could trick the contract into transferring the incorrect amount of funds by modifying the decimal count of the deposited asset.

This would require the attacker to have control over the base asset, as well as create two matching orders that the execution facility agrees to fill.

Exploit Scenario:

A malicious actor puts in a buy limit for 1000 units of asset A, for a max price of 100 asset B each.

1. As part of the order, the malicious actor must deposit 100 units of asset B
2. Both assets A and B are set to 8 decimals at this point.
3. After the order is created and the quote asset B has been deposited, asset A's decimals now change to 4 .

Now, if the order is partially filled, e.g the execution facility triggers an exchange of 1 unit of asset A for the quoted price in asset B - the code will recalculate the price for that single unit in the new decimal system, resulting in a payout that is 4 orders of magnitude greater than the deposit that was initially provided, thus allowing the attacker to transfer more than the initial deposit, causing a net loss to the contract.

Recommendation:

This could be resolved by storing the decimals count at the time of deposit as part of the order. However, since this is not in any way a legitimate or commonplace behavior - we recommend that the contract acts defensively and refuses to interact with unfamiliar assets. For this purpose, a whitelist could be implemented so that only known and trusted assets are dealt with.

Customer's response:

Acknowledged. Asset whitelist implemented via supported asset checks when creating and filling orders.

Commit: 8c98f30d8a59defe7ab087f5beff251cd865e600

https://github.com/ShiftForex/cables_stellar_contracts/tree/8c98f30d8a59defe7ab087f5beff251cd865e600

Fix review: OK

## M-3. Centralization risk

Description:

As a more general design issue - we believe this contract has significant exposure to potentially malicious influence due to the fact that there are no guarantees regarding the behavior of the off-chain components, which may act in a way that does not necessarily benefit the end users.

For example - an execution facility may prioritize or ignore certain users orders without necessarily accounting for that behavior in a fair or transparent manner.

Moreover, since filling orders is done in an ordered manner - i.e. there is a maker and taker, the execution facilities may behave in a way that prioritizes fee collection rather than optimizing for maximum user value.

Customer's response:

Acknowledged.

## Low Severity Issues

### L–1 Fee avoidance when an asset has a low decimal count

Description:

Since the fee is computed as 1% of the transaction value - which is a computation that requires a minimum of 3 decimal places - transactions for any asset that has lower than 3 decimals will avoid paying the fee.

Recommendation: creating a whitelist of valid assets - will also resolve this issue.

Customer's response:

Acknowledged. Asset whitelist implemented via supported assets checks when creating and filling orders.

Commit: 8c98f30d8a59defe7ab087f5beff251cd865e600

https://github.com/ShiftForex/cables_stellar_contracts/tree/8c98f30d8a59defe7ab087f5beff251cd865e600

Fix review: OK

## L–2 Incorrect validation of transfer amount in move_token

Description:

The function move_token is a utility function to transfer funds in a given asset between two addresses.
The function intends to validate that the transfer amount is positive, but in actuality, it only verifies that it is non-zero:

```
fn move_token(env: &Env, token: &Address, from: &Address, to: &Address, transfer_amount: i128) {
    // require transfer_amount > 0
    if transfer_amount == 0 {
        panic_with_error!(&env, Error::ZeroValue);
    }
```

Recommendation:

We recommend correcting this so the code matches the comment, and prevents non-positive values from being passed into the underlying transfer function.

Customer's response:

Acknowledged. Recommendation implemented.

Commit: ba23eea806f582669765af21ea1408c8ab318d9a

https://github.com/ShiftForex/cables_stellar_contracts/tree/ba23eea806f582669765af21ea1408c8ab318d9a

Fix review: OK

## Informational Severity Issues

### INFO–1 Dead-end flow when iterating through list of open orders

Description:
Due to the way resource limits work in Soroban and in other smart-contract platforms - any loops that have a dynamic amount of iterations (e.g. traversing the list of active execution facilities) will hit the resource limit at some point.

This is manifested in `close_order`, where the list of open orders for a user is traversed using binary search. If a given user has a large enough number of orders, the list size will be too large to traverse fully before exhausting the resource limit, thus causing the call to always fail, making it impossible to realize certain flows in the contract.

Recommendation:

While it is extremely unlikely that this issue could occur in a real use case, we recommend that the maximum amount of open orders be capped explicitly in the code, for posterity and increased robustness.

Customer's response:

Acknowledged. Max number of open orders hardcoded into the contract (1,000). Check and error implemented for new order creation.

Commit: 2a90646eaf0b009b67a97fe3961facea88b43cca

https://github.com/ShiftForex/cables_stellar_contracts/tree/2a90646eaf0b009b67a97fe3961facea88b43cca

Fix review: OK

### INFO–2 Dead-end flow when iterating through list of execution facilities

Description:
Due to the way resource limits work in Soroban and in other smart-contract platforms - any loops that have a dynamic amount of iterations (e.g. traversing the list of active execution facilities) will hit the resource limit at some point.

Similar to CBLS-003 - this issue can occur if the list of execution facilities becomes too large - and consistently prevents certain flows in the contract from completing before hitting the resource limit.

Recommendation:

While it is extremely unlikely that this issue could occur in a real use case, we recommend that the maximum amount of execution facilities be capped explicitly in the code, for posterity and increased robustness.

Customer's response:

Acknowledged. Max number of active facilities hardcoded into the contract (100). Check and error implemented for setting and reactivating facilities.

Commit: 2a90646eaf0b009b67a97fe3961facea88b43cca

https://github.com/ShiftForex/cables_stellar_contracts/tree/2a90646eaf0b009b67a97fe3961facea88b43cca

Fix review: OK

## INFO-3 Redundant slippage field in orders

Description:
Currently, market orders have a slippage field that is passed in as part of the user's order parameters. However, this field is not utilized anywhere.

Recommendation:

We recommend either enforcing slippage in a manner relevant to the behavior of the execution facilities or alternatively removing this field from the code.

Customer's response:

Acknowledged. Field removed.

Commit: 27e620a68bdfcabd91a1a5ef630e86619956dd52

https://github.com/ShiftForex/cables_stellar_contracts/tree/27e620a68bdfcabd91a1a5ef630e86619956dd52

Fix review: OK

## INFO-4 Incorrect error (currency matching) for pricing validation

Description:

In the exchange function, in cases where the buy orders limit price is lower than the sell orders limit price (meaning the two orders do not match) - the function will return an error indicating mismatched assets, rather than indicating the pricing issue.

Recommendation:

We recommend separating the tests such that the correct error is returned.

Customer's response:

Acknowledged. New errors added for pricing and order type validation.

Commit: fedcdef55f43bc4a79aab830695f568aabc33d96

https://github.com/ShiftForex/cables_stellar_contracts/tree/fedcdef55f43bc4a79aab830695f568aabc33d96

Fix review: OK

## INFO-5 Code duplication for get_active_facilities, get_execution_facility, is_execution_facility

Description:

The `get_active_facilities`, `get_execution_facility`, `is_execution_facility` functions are not used internally in the `exec.rs` code, instead the call to the env storage API is duplicated across all places which require these lists.

Customer's response:

Acknowledged. Ledger reading and storing optimized.

Commit: 9fa45316675d8a58d13a9c485cdb4642562bc2b9

https://github.com/ShiftForex/cables_stellar_contracts/tree/9fa45316675d8a58d13a9c485cdb4642562bc2b9

Fix review: OK

# Disclaimer

The Certora Prover takes a contract and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification and the Certora Prover does not check any cases not covered by the specification.

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

# About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.