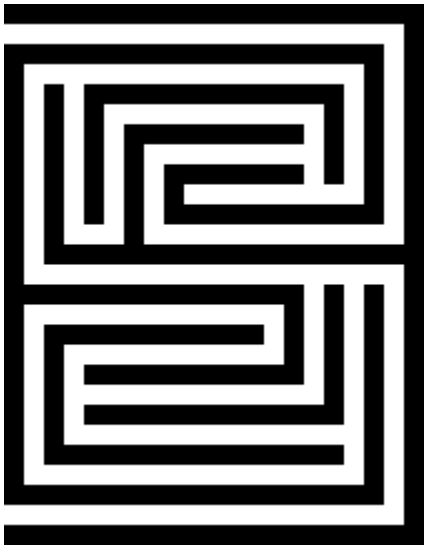

Saber.so Public Report

PROJECT: Saber.so Audit

May 2021

Prepared For:

Saber

team@saber.so

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Saber.so Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in May of 2021 to perform a comprehensive security review of the Saber.so smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of two business days by both members of the Bramah Systems, LLC. executive staff.

Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Timeline

Review Commencement: May 14, 2021

Report Delivery: May 19th, 2021

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Saber.so protocol, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Rust code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Specification was provided in the form of code comments. The contracts were provided via GitHub (commit hash **d811ec1fc072d60aad66cbd006c8ba19e21f165b**)

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the Saber.so protocol. During the course of our engagement, Bramah Systems found minimal instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT. The team has since addressed these issues with a resolution or risk acceptance.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the Saber.so Protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Rust compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the Saber.so protocol or any other relevant product, service or asset of Saber.so or otherwise. This report is not and should not be relied upon by Saber.so or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the Saber.so Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.

General Recommendations

Best Practices & Rust Development Guidelines

Usage of Structs suggested

Throughout the protocol, there's numerous statements that could be simplified (**adminFee, tokenMint, tokenAccount**) which could be simplified within structs (specific to token A and token B, for example).

Resolution: The Saber.so team modified the swap state to move swap token state into a shared struct and modified several functions to use this swap state.

Deprecated function usage via create_account

As per the Solana SDK, usage of **create_account** is deprecated since 1.5.17: one should use **create_account_for_test** instead.

Resolution: The Saber.so team replaced all uses of create_account with create_account_for_test.

Usage of unwrap presents concern

Because the **unwrap** function may panic, its use is generally discouraged. Instead, we suggest using pattern matching and handling the **Err** case explicitly, or calling **unwrap_or**, **unwrap_or_else**, or **unwrap_or_default**.

Resolution: The Saber.so team removed all usages of unwrap() in the program/ directory.

Specific Recommendations

Unique to the Saber.so Protocol

Usage of magic numbers referenced in source code

The following line is present within **fees.rs** (line 68) and should be decided upon prior to deployment to a wider user base.

```
.checked_div((n_coins.checked_sub(1)?).checked_mul(4)?)?; // XXX: Why divide by 4?
```

Resolution: The Saber.so team added a comment explaining the reasoning by the division by 4.

TODO Item remains in source

The following line is present within **instruction.rs** (line 793) and should be decided upon prior to deployment to a wider user base.

```
/// TODO actually pack / unpack instead of relying on normal memory layout.
```

Resolution: The Saber.so team resolved all TODOs in the codebase.

Typographic errors in code comments

We heavily recommend a spelling linter be used upon the various code comments within the protocol. Instances include:

```
minimum_b_amount * 30, // XXX: 10 -> 30; Revisit this splippage multiplier
```

```
// slippage exceeeded: minimum out amount too high
```

Resolution: The Saber.so team ran a spell checker through the code.

Toolset Warnings

Unique to the Saber.so Protocol

Overview

In addition to our manual review, our process involves utilizing static analysis and formal methods in order to perform additional verification of the presence of security vulnerabilities (or lack thereof). An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable.

Compilation Warnings

No compilation warnings were encountered during the course of our audit.

Test Coverage

The contracts possess a number of functional unit tests encompassing various stages of the application lifecycle.

Static Analysis Coverage

The contract repository underwent scrutiny with static analysis agents, but largely benefits from the inherent security posture that Rust provides. Rust statically enforces many properties of a program beyond memory safety, including null pointer safety and data race safety. With a well defined test suite and ample documentation as to the functionality, the Saber.so contracts go into deep detail as to how each implementation functions.

Directory Structure

At time of review, the directory structure of the Saber.so smart contracts repository appeared as it does below. Our review, at request of Saber.so, covers the Rust code (*.rs) as of commit

hash **d811ec1fc072d60aad66cbd006c8ba19e21f165b**

```
.
├── Cargo.toml
├── Xargo.toml
├── proptest-regressions
│   └── curve.txt
└── src
    ├── admin.rs
    ├── bn.rs
    ├── curve.rs
    ├── entrypoint.rs
    ├── error.rs
    ├── fees.rs
    ├── instruction.rs
    ├── lib.rs
    ├── pool_converter.rs
    ├── processor.rs
    ├── state.rs
    └── utils.rs

2 directories, 15 files
```