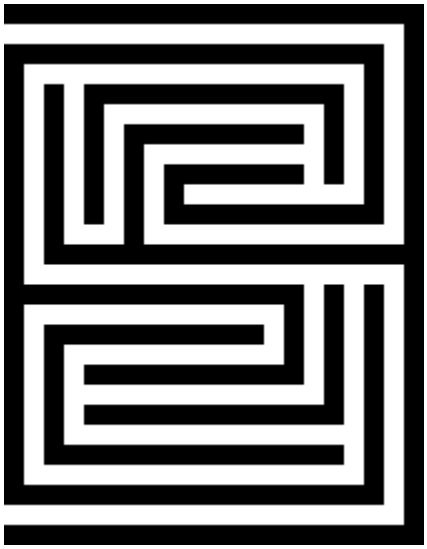

Stabilize Finance Public Report

PROJECT: Stabilize Finance

February 2021

Prepared For:

Stabilize Finance

hello@stabilize.finance

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Stabilize Finance Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in February of 2021 to perform a comprehensive security review of the Stabilize Finance smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of three days by a member of the Bramah Systems, LLC. executive staff.

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

Timeline

Review Commencement: February 5th, 2021

Report Delivery: February 9th, 2021

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Stabilize Finance 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 Solidity code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Contract specification was provided in the form of code comments and functional unit tests, along with a verbose specification document which provided justification for infrastructure decisions and structural fundamentals.

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the Stabilize Finance Protocol. During the course of our engagement, Bramah Systems found multiple instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT, as our report details.

This noted, the team used reviewed and vetted components (primarily from OpenZeppelin) and provided details as to their intent in which differentiations existed between best practice and the team's implementation, which helped Bramah highlight any potential concerns with their approach.

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 Stabilize Finance 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 Solidity 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 Stabilize Finance protocol or any other relevant product, service or asset of Stabilize Finance or otherwise. This report is not and should not be relied upon by Stabilize Finance 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 Stabilize Finance 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 & Solidity Development Guidelines

Hard-coded addresses in source code

Multiple hard-coded addresses without setter methods exist within the source code. In the event that these external addresses are changed (given they are those of an external party, this is not outside of the realm of possibility), lack of setter functions will result in functionally inconsistent operating of the Stabilize Finance protocol.

Resolution:The team has provided the following: “Those contracts that contain hard-coded addresses can be changed by the team when needed and pointers to those contracts updated as well.”
Bramah concurs that the contracts can be changed, but doing so may be considered disruptive (nor does our audit extend to these changes or any future contracts).

Pre-flattened Solidity files

The Solidity files come flattened, containing all contracts needed for deployment of the contract. However, by deploying this way, comparing changes between files (especially those considered to be template code) becomes increasingly difficult, which may present confusion.

Resolution: The team has provided the following:
“During development, our team has found this method of contract flattening to be easier to read by the development team.”

Excess gas consumption resulting from .length usage in loop

The usage of **.length** as the upper bound of “for loops” is not suggested for large array sizes, as holding its value in a local variable is more gas efficient.

File: contracts-master/contracts/strategies/StabilizeStrategyBACMICArb.sol

Lines: 667-671

File: contracts-master/contracts/strategies/StabilizeStrategyBACMICArb.sol

Lines: 865-874

File: contracts-master/contracts/strategies/StabilizeStrategyFRAXArb.sol

Lines: 655-659

File: contracts-master/contracts/strategies/StabilizeStrategyFRAXArb.sol

Lines: 713-722

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArbV2.sol

Lines: 670-674

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArbV2.sol

Lines: 803-815

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV3.sol

Lines: 730-737

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV3.sol

Lines: 673-677

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV3.sol

Lines: 806-813

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV3.sol

Lines: 783-793

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArb.sol

Lines: 800-812

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArb.sol

Lines: 668-672

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 728-735

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 826-833

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 748-755

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 784-794

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 674-678

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV4.sol

Lines: 804-811

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArbV3.sol

Lines: 668-672

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArbV2.sol

Lines: 960-965

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArbV2.sol

Lines: 750-759

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArbV2.sol

Lines: 694-698

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArb.sol

Lines: 746-755

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArb.sol

Lines: 965-970

File: contracts-master/contracts/strategies/StabilizeStrategyStablecoinArb.sol

Lines: 686-690

File: contracts-master/contracts/strategies/StabilizeStrategyDSDESDArb.sol

Lines: 973-983

File: contracts-master/contracts/strategies/StabilizeStrategyDSDESDArb.sol

Lines: 677-681

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV2.sol

Lines: 725-735

File: contracts-master/contracts/strategies/StabilizeStrategyBTCArbV2.sol

Lines: 669-673

File: contracts-master/contracts/zs-BMSGR.sol

Lines: 957-961

File: contracts-master/contracts/Treasury.sol

Lines: 608-610

File: contracts-master/contracts/Treasury.sol

Lines: 639-645

File: contracts-master/contracts/strategies/StabilizeStrategySeigniorageArbV3.sol

Lines: 860-870

File: contracts-master/contracts/zs-SGR.sol

Lines: 966-970

File: contracts-master/contracts/zs-USD.sol

Lines: 986-990

File: contracts-master/contracts/zs-USD.sol

Lines: 1028-1033

File: contracts-master/contracts/Operator.sol

Lines: 691-694

File: contracts-master/contracts/Operator.sol

Lines: 716-719

File: contracts-master/contracts/zs-BTC.sol

Lines: 966-970

File: contracts-master/contracts/StabilizeTornadoProxyV2.sol

Lines: 881-883

File: contracts-master/contracts/zs-FRAX.sol

Lines: 962-966

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Sensitive parameter changing functions should emit an event

Starting with **startGovernanceChange** and applying to each function thereafter, multiple functions within the **Operator.sol** contract allow material modifications to sensitive parameters (e.g. those which impact the most basic operation of the protocol). It is heavily suggested that these functions emit an event on invocation for the sake of overall transparency.

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

External is preferable to public for gas optimization

As noted by multiple other static code analysis tools, the usage of external function visibility is preferable to public, in that external functions are prevented from being called internally, whereas public functions can be called internally. This results in a gas optimization that is due to the fact that Solidity copies arguments to memory on a public function whereas external functions read from calldata (which is cheaper than memory allocation)

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Compiler version (pragma) not locked

Throughout the repository the compiler pragma is not locked, allowing for any version of Solidity at or above that version (denoted by a ^) to compile the contracts. As future versions may change language constructs and assumptions that exist within this version of the codebase, it is suggested that the unlocked pragma be modified.

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Variable naming idiosyncrasies

In multiple for loops throughout the contract repository, the usage of the variable “**i2**” is used in place of a second iterator. As usage of **i2** is rather unclear to its immediate purpose, we suggest the usage of a different letter or iterator (such as **j** or **k**)

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Constants should be SNAKE_CASE

In order to be compliant with the Solidity style guide, in instances where constants are utilized,

they should be represented in snake case.

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Usage of send and transfer considered against best-practice

Following EIP-1884, the usage of **transfer and send** is no longer suggested, due to changing gas costs in their usage. Use **.call.value(...)("")** instead. The contract presently makes use of **transfer** throughout.

Resolution: The Stabilize Finance team provided the following: “Where possible, we will consider this in future contract deployments.”

Specific Recommendations

Unique to the Stabilize Finance Protocol

Highly permissive owner account and centralization of power

The deploying account possesses a number of highly actions (namely, initiating per transaction burning). This deploying account should (where possible) minimize usage of the associated key (e.g. performing transactions, using as a regular user account) and perform other operational security best practices. Potentially, this could involve transferring ownership to a MultiSignature governance.

Resolution: The team provided the following: “The contracts utilize 24 hour timelocks to limit the speed at which governance can change the contracts. This gives users a window of time to withdraw funds before changes are implemented.”

Bramah confirms the existence of the timelocks, but still cautions regarding the centralization of power.

Usage of magic-numbers to be avoided

As gas prices (and the gas required for certain actions) has and will continue to change over

time, it is important to not hardcode in any values which materially affect performance of the protocol.

> **uint256 gasPayoutEstimate = startGas.sub(gasleft()).add(60000); // Estimate the amount of gas used by this operation and the next few**

Resolution: The Stabilize Finance team provided the following: "In the future, the team plans to implement setters for values that can be changed in future."

Wei conversion logic should be a function (D.R.Y. best practice)

Logic for Wei conversion is utilized multiple times throughout the contract repository. Rather than having multiple instances of identical code in disparate locations, we suggest creating a singular function that is called for this calculation.

Resolution: The Stabilize Finance team provided the following: "We will consider this in future contract deployments."

Usage of tx.origin to determine if sender is a smart contract

The contract utilizes transaction.origin in multiple places to determine whether or not the sender is a smart contract. As this would block any protocol participant from utilizing a multi-signature wallet to interact with the protocol, we suggest this code-block be removed.

Resolution: The Stabilize Finance team provided the following: "As we continue to evaluate our user-base and advance our research into protections against malicious contracts interacting with our protocol, we will implement ways to integrate all interactions with our contracts while protecting our users."

Violations of checks-effects-interactions throughout

Throughout the protocol, but extensively within **Operator.sol** (specific functions denoted below) there exists multiple code-patterns which suggest potential re-entrancy. Where possible, a mechanism such as **ReentrancyGuard** should be implemented to avoid potential exploitation of this design flaw.

Operator.mintNewWeek(): In violation

Operator.rebalancePoolRewards(): In violation

Operator.pushReward(uint256,address): In violation

Operator.getReward(uint256): In violation

Operator.bootstrapLiquidty(): In violation

Resolution: The team provided the following responses:

- Regarding the re-entrancy possibility in Operator.rebalancePoolRewards(), the function updates the time after the checks, we believe preventing any potential re-entrancy from going beyond the first few lines
- Regarding the re-entrancy possibility in Operator.pushReward(uint256,address) and Operator.getReward(uint256), the functions update the earned reward to 0 before calling any external functions, we believe preventing any potential re-entrancy from having an effect
- Regarding the re-entrancy possibility in Operator.mintNewWeek(), the external contract call prior to check-effects is to a trusted contract that cannot be changed (the Stabilize token contract), we believe preventing any potential re-entrancy from subsequent external contract calls from having an effect.

Bramah has independently confirmed each response, and while there is a potential for reentrancy for each (as noted above), these mitigations as discussed by the team would sufficiently mitigate the concern for the listed mitigations above. This notably does not extend to functions not referenced above.

Unclear magic number usage in weighting

The following line is present within the **rebalancePoolRewards** function:

uint256 weightReduction = diff.mul(50); // Weight is reduced for each $0.0001 above target price

It is unclear what this magic number of 50 aims to achieve or how and why 50 was chosen as an appropriate weighting.

Resolution: The Stabilize Finance team provided the following: “This value is an arbitrary number chosen by the team to balance out rewards rates among the pools. In the future, we plan to implement new ways to distribute rewards among our stablecoin pools.”

Toolset Warnings

Unique to the Stabilize Finance 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 warnings were present at time of compilation.

Test Coverage

The contract repository possesses extensive unit test coverage throughout. This testing provides a variety of unit tests which encompass the various operational stages of the contract.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Securify
- MAIAN
- Mythril
- Oyente
- Slither

In each case, the team had either responded the concern above, mitigated the concern raised or provided adequate justification for the risk (such as adhering to the ERC-20 standard).

Directory Structure

At time of review, the directory structure of the Stabilize Finance smart contracts repository appeared as it does below. Our review, at request of Stabilize Finance, covers the Solidity code (*.sol) as of commit-hash **e5dc484** of the Stabilize Finance repository.

.

```
├── LICENSE
├── README.md
├── contracts
│   ├── GasTreasury.sol
│   ├── Operator.sol
│   ├── PriceOracle.sol
│   ├── PriceOracleV2.sol
│   ├── PriceOracleV3.sol
│   ├── StabilizeStakingPool.sol
│   ├── StabilizeToken.sol
│   ├── StabilizeTornadoProxy.sol
│   ├── StabilizeTornadoProxyV2.sol
│   ├── Treasury.sol
│   ├── strategies
│   │   ├── StabilizeStrategyBACMICArb.sol
│   │   ├── StabilizeStrategyBTCArbV2.sol
│   │   ├── StabilizeStrategyBTCArbV3.sol
│   │   ├── StabilizeStrategyBTCArbV4.sol
│   │   ├── StabilizeStrategyDSDESDArb.sol
│   │   ├── StabilizeStrategyFRAXArb.sol
│   │   ├── StabilizeStrategyPickle.sol
│   │   ├── StabilizeStrategyPickleDAI.sol
│   │   ├── StabilizeStrategySeigniorageArb.sol
```

```
│   │   ├── StabilizeStrategySeigniorageArbV2.sol
│   │   ├── StabilizeStrategySeigniorageArbV3.sol
│   │   ├── StabilizeStrategyStablecoinArb.sol
│   │   └── StabilizeStrategyStablecoinArbV2.sol
│   ├── za-DAI.sol
│   ├── za-USDC.sol
│   ├── za-USDT.sol
│   ├── za-sUSD.sol
│   ├── zs-BMSGR.sol
│   ├── zs-BTC.sol
│   ├── zs-DAI.sol
│   ├── zs-FRAX.sol
│   ├── zs-SGR.sol
│   └── zs-USD.sol
└── docs
    └── whitepaper.pdf

3 directories, 36 files
```