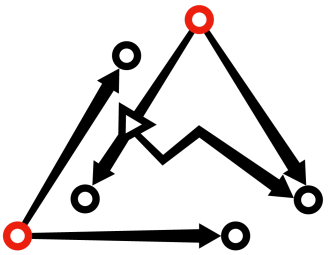

Least Authority

PRIVACY MATTERS

Multisafe

Security Audit Report

Trust Machines

Initial Audit Report: 9 November 2022

This Audit Report is intended for internal use and discussion purposes only. We advise against sharing this report beyond trusted team members and recommend that publication take place only after the verification has been completed and the Final Audit Report has been delivered.

Table of Contents

Overview

Background

Trust Machines have requested that Least Authority perform a security audit of the Multisafe, a shared multi-signature crypto wallet for managing Stacks (STX) and Bitcoin (BTC).

Project Dates

- **October 4 - October 12:** Initial Code Review *(Completed)*
- **October 13:** Delivery of Initial Audit Report *(Completed)*
- **November 3:** Verification Review *(Completed)*
- **November 9:** Delivery of Final Audit Report *(Completed)*

Review Team

- Xenofon Mitakidis, Security Researcher and Engineer
- Mirco Richter, Cryptography Researcher and Engineer
- ElHassan Wanas, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of Multisafe followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repositories are considered in scope for the review:

- Multisafe:
  https://github.com/Trust-Machines/multisafe
  - `/safe.clar`
  - `/traits.clar`
  - `/executors/add-owner.clar`
  - `/executors/allow-caller.clar`
  - `/executors/magic-bridge-send.clar`
  - `/executors/magic-bridge-set.clar`
  - `/executors/remove-owner.clar`
  - `/executors/revoke-caller.clar`
  - `/executors/set-threshold.clar`
  - `/executors/transfer-sip-009.clar`
  - `/executors/transfer-sip-010.clar`
  - `/executors/transfer-stx.clar`
  - `/helper/ft-none.clar`
  - `/helper/nft-none.clar`

Specifically, we examined the Git revision for our initial review:

`19391da588c6231ebf4e604e2ab65c45a7dc7ddf`

For the review, this repository was cloned for use during the audit and for reference in this report:

Multisafe:
https://github.com/LeastAuthority/trustmachines-multisafe

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- Multisafe Audit: https://github.com/Trust-Machines/multisafe/blob/main/audit/Trust-Machines-MS-Audit(2022-07).pdf

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation;
- Adversarial actions and other attacks on the network;
- Potential misuse and gaming of the smart contracts;
- Attacks that impacts funds, such as the draining or manipulation of funds;
- Mismanagement of funds via transactions;
- Denial of Service (DoS) and other security exploits that would impact the intended use of the smart contracts or disrupt their execution;
- Vulnerabilities in the L1 and smart contracts code;
- Protection against malicious attacks and other ways to exploit the L1 and smart contracts code;
- Inappropriate permissions and excess authority;
- Data privacy, data leaking, and information integrity; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Multisafe is designed to enable up to 20 users to control a Stacks (STX) and Bitcoin (BTC) shared wallet. Multisafe is configurable and implements functionality that users can perform individually, in addition to functionality that must be approved by a threshold number of users in the group. The Multisafe system is composed of a frontend web application and a smart contract suite implemented in Clarity. The scope of this review was limited to the review of the design and implementation of the smart contract component.

We analyzed the implementation with a particular emphasis on ways in which a malicious smart contract could be implemented for the `executor` trait, but did not identify any vulnerabilities. In addition, we checked for edge cases that would lead to unintended smart contract behavior but did not identify any issues.

Our review was supplemented by a previous audit of the system by a third-party security team.

System Design

Our team performed a close review of the design of the on-chain component of the Multisafe system and found that the Multisafe team has taken security into consideration. In our investigation of the `executor` trait and the Multisafe contract, we found that we agree with the Multisafe team's concern that this design

potentially increases the attack surface because a smart contract that implements this trait can be programmed to execute almost arbitrary behavior. However, we were not able to identify any specific attacks.

We checked the various implementations of the `executor` trait in `main/contracts/executors`, looking for potential replay or reentrancy attacks but did not identify any issues.

Additionally, we checked for cases where `tx-sender` could be used for access control and did not identify any vulnerabilities. While the usage of `as-contract` does not affect `tx-sender`, it does change the value of `tx-sender` that subsequent calls can access, effectively changing `tx-sender` during the execution of the contract call. According to the clarity documentation, `tx-sender` should be the original principal that signs the transaction. However, the `safe` contract relies on being able to change `tx-sender` to create the intended behavior, and this is a security concern.

Code Quality

In our review, we found that the codebase is very well organized and written. However, we found that parameter names of implemented `executors` could be improved (Suggestion 1).

Tests

The unit and integration tests in place provide sufficient test coverage of the smart contracts. However, there are missing success case tests for `mb-initialize-swapper` and `mb-escrow-swap`. We recommend increasing the unit test coverage to make it more comprehensive (Suggestion 2).

Documentation

The documentation provided for this security audit was sufficient in describing the functionality of the Multisafe smart contracts. Our team found that the documentation facilitated the understanding and assessment of the expected behavior of the system.

Code Comments

Our team found the code to be well commented and in accordance with best practices.

Scope

The scope of this security review included all the on-chain security-critical components. However, the system is supported by an off-chain component, which was assumed to function as intended for the purpose of this review. We recommend that off-chain components be reviewed in a future security audit.

Dependencies

The Multisafe smart contracts depend on the correct implementation and functionality of Magic Bridge. The Magic Bridge implementation was reviewed by our team, and the Initial Audit Report, in which issues and suggestions were identified, was delivered on September 30, 2022. However, as of the writing of this report, the findings from that review have not been addressed and verified.

The security of the Multisafe smart contracts is critically dependent on the security and correct functionality of the `executor` trait. As a result, we recommend that any implementation of that trait be audited.

Specific Issues & Suggestions

We list the issues and suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

ISSUE / SUGGESTION	STATUS
Suggestion 1: Improve Parameter Names in the Base Executors	Unresolved
Suggestion 2: Improve Unit Test Coverage	Unresolved

Suggestions

Suggestion 1: Improve Parameter Names in the Base Executors

Location

`contracts/executors/add-owner.clar`

`contracts/executors/allow-caller.clar`

`contracts/executors/magic-bridge-send.clar`

And other contracts under:

`contracts/executors`

Synopsis

Currently, specific `executor` implementations have vague parameter names like `param-u` and `param-p`. This reduces the readability of the implementation of those `executors`.

Mitigation

For unused parameters to an `executor`, the names used can be kept as is. However, if a parameter is used in the implementation of an `executor` function, then we recommend giving it a meaningful name, such as `new-owner` instead of `param-p`.

Status

The Multisafe team prefers to keep the naming scheme used in the parameters for consistency.

Verification

Unresolved.

Suggestion 2: Improve Unit Test Coverage

Location

`contracts/safe.clar`

`contracts/safe.clar`

Synopsis

While the test coverage is nearly at 100%, the Magic Bridge related functionality in the Multisafe contract is not tested for success cases.

Mitigation

We recommend adding tests to ensure that `mb-initialize-swapper` and `mb-escrow-swap` succeed, given a real Magic Bridge contract.

Status

To avoid complicating the testing setup, the Multisafe team has decided against adding an integration test with a real Magic Bridge contract. However, they have indicated that they verified the interaction manually before deployment and will continue to do so with further updates.

Verification

Unresolved.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, and zero-knowledge protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.