Least Authority
PRIVACY MATTERS

Keep Network Solana Smart Contracts
Security Audit Report

Thesis

Final Audit Report : 29 August 2023

Table of Contents

Overview

Background

Keep Network has requested that Least Authority perform a security audit of their Solana Smart contracts, in preparation for the upcoming deployment of the contracts on the Solana Network.

Project Dates

- **August 8, 2023 - August 11, 2023:** Initial Code Review *(Completed)*
- **August 15, 2023:** Delivery of Initial Audit Report *(Completed)*
- **August 29, 2023:** Verification Review *(Completed)*
- **August 29, 2023:** Delivery of Final Audit Report *(Completed)*

The dates for verification and delivery of the Final Audit Report will be determined upon notification from the Keep Network team that the code is ready for verification.

Review Team

- Nicole Ernst, Security Researcher and Engineer
- Mukesh Jaiswal, Security Researcher and Engineer
- Xenofon Mitakidis, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Keep Network Solana Smart Contracts followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repository is considered in scope for the review:

- Solana Smart Contracts:
  https://github.com/keep-network/tbtc-v2/tree/main/cross-chain/solana

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

- `b01aca102a162bb6d3768db59b0811988498a55d`

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

- Solana Smart Contracts:
  https://github.com/LeastAuthority/Solana-Smart-Contracts/tree/main/cross-chain/solana

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:

- Website:
  https://keep.network

- RFC 8: Cross-chain Tokenized Threshold BTC: https://github.com/keep-network/tbtc-v2/blob/main/docs/rfc/rfc-8.adoc

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 smart contracts' code;
- Protection against malicious attacks and other ways to exploit the smart contracts;
- Inappropriate permissions and excess authority;
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Our team performed a security audit of the Solana Smart Contracts and their core components, the `tBTC` and `wormhole-gateway` contracts, which aim to facilitate the use of tBTC within the Solana Network. The gateway contract allows the Wormhole to mint a tBTC token on the Solana Network, which represents the locked tBTC on Ethereum. Our team reviewed the Keep Network Solana Smart Contracts implementation for errors and security vulnerabilities and could not identify any issues in the implementation and design of the smart contracts.

We found that the system is very well-designed, and that the coded implementation is well-organized. The Keep Network team has taken security into consideration in the design and development of the Solana Smart Contracts as demonstrated by the usage of the Anchor Framework, which results in a more robust and secure system.
In our review, our team found that accounts that are already guardians can be added as guardians multiple times. Although this does not currently pose any security concerns, we still recommend preventing this functionality to avoid unintended behavior in future versions of the implementation (Suggestion 1).

Documentation

The project documentation provided to our team in the form of the RFC and the documentation within the code provided an accurate and helpful general overview of the system design.

Code Comments

We found sufficient code comments describing the intended behavior of security-critical functions and components.

Scope

The scope of this review was sufficient and included all security-critical components.

Dependencies

Our team did not identify any vulnerabilities in the implementation's use of dependencies.

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: Prevent Guardian From Being Added Multiple Times	Resolved

Suggestions

Suggestion 1: Prevent Guardian From Being Added Multiple Times

Location

`processor/admin/add_guardian.rs#L46-L64`

Synopsis

Accounts that are already guardians can be added as guardians again, which could potentially lead to unintended behavior if future additions are implemented in the system.

Mitigation

We recommend deduping the vector storing the guardians after adding a guardian and setting the number of guardians to the length of the vector afterwards.

Status

The Keep Network team has implemented a Multi-Sig and transferred the ownership of the contracts to the Threshold Council.

Verification

Resolved.

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, zero-knowledge protocols, and consensus 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.