Security Audit Report

Structured Private Deposit Whitelist

v1.0

December 4, 2025

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has been engaged to perform a security audit of Structured Private Deposit Whitelist.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/structured-org/spdbtc
Scope	The scope is restricted to the changes applied in the following pull requests: • https://github.com/structured-org/spdbtc/pull/16 reviewed at commit `e44bab2267853095948055368 4c00013e6caaecf`, base branch at `a1050c143984aee1caa18a840fd5d46a0166664d`.
Fixes verified at commit	`0337a515c5fe1292518119403eb0a2c1814e70a3`

	Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The Structured Private Deposit smart contract is an ERC-4626 compliant vault that enables users to deposit WBTC (Wrapped Bitcoin) and receive spdBTC tokens at a 1:1 ratio.

The contract features a custodial model where deposited WBTC is transferred to a designated custodian address. It includes essential security measures such as deposit limits, address blacklisting capabilities, and a pause mechanism.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending**, **Acknowledged**, **Partially Resolved**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Low-Medium	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	The contract is well documented.
Test coverage	Medium	`hardhat coverage` reports a `74.44%` test coverage.

Summary of Findings

No	Description	Severity	Status
1	Inconsistent whitelist and blacklist state enables partial token locking	Minor	Resolved
2	Missing public view function to query whitelist status	Informational	Resolved
3	Miscellaneous comments	Informational	Resolved

Detailed Findings

1. Inconsistent whitelist and blacklist state enables partial token locking

Severity: Minor

In `contracts/spdBTC/spdBTC.sol:477-490`, the `setWhitelisted` function allows contract owners to mark addresses as whitelisted independently of their blacklist status.

However, no mutual exclusivity is enforced between the two states, allowing deposits to succeed even when the address is also blacklisted.

This results in a situation where spdBTC is minted and credited to a blacklisted address, which can subsequently prevent token transfers or withdrawals due to blacklist enforcement in the contract.

Recommendation

We recommend enforcing mutual exclusivity between the whitelist and blacklist.

Status: Resolved

2. Missing public view function to query whitelist status

Severity: Informational

The whitelist feature was added to the `SpdBTC` contract, but there is no public view function to let users or off-chain services check whether an address is whitelisted.

Without a getter (similar to the `isBlacklisted` function), callers must read storage directly from the chain or rely on events, which is inconvenient and error-prone. This reduces UX for users and integrators.

Recommendation

We recommend adding a public view function that returns whether an address is whitelisted and adding the function signature to the interface. Ensure the chosen function name does not conflict with the existing `isWhitelisted` modifier.

Status: Resolved

3. Miscellaneous comments

Severity: Informational

The codebase contains several maintenance and clarity issues:

- The parameter documentation in `contracts/spdBTC/spdBTC.sol:481`: *"@param `_isWhiteisted` Whether to blacklist or unblacklist the address."* incorrectly mentions blacklist instead of whitelist. Also, `_isWhiteisted` has incorrect spelling.
- The comment in `contracts/spdBTC/spdBTC.sol:118` claims the whitelist is required for both deposit and token transfer functionalities, which is inconsistent with the actual functionality.

Recommendation

We recommend:

- Replacing "(un-)backlist" with "(un-)whitelist", and `_isWhiteisted` with `_isWhitelisted`.
- Editing the comments to describe intended functionality.

Status: Resolved