Cryptonics

Continuous Audit Report

pTokens Bridges

SEPTEMBER 16, 2020

Table of Contents

Disclaimer

This audit has been performed by

Philip Stanislaus

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/
info@cryptonics.consulting

Summary of Findings

No	Description	Severity	Status
1	EOS block submission does not enforce submission of blocks with schedule/producer set changes	Major	Acknowledged
2	EOS block validation in core-private does not check whether block has been confirmed by enough producers	Medium	Acknowledged
3	Disabling validation when not in debug mode	Medium	Resolved
4	Linear memory increase of tx ids list	Minor	Resolved
5	Nightly toolchain	Minor	Resolved
6	Code duplication	Minor	Acknowledged
7	DB key collisions	Minor	Resolved
8	Bridge requires EOS names with more than 1 character	Minor	Resolved
9	Output of debugging could accidentally trigger slashable behavior	Minor	Resolved

This report contains 9 findings on 12 pages (including one cover page).

All issues encountered during the audit process have been addressed by the team.

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a continuous audit of the pTokens 2-way asset transfer bridges, forming part of the pTokens project (https://ptokens.io/). The engagement is ongoing and includes the changes performed to the codebase since the last audit report from April 24, 2020.

The objectives of the audit are as follows:

1. Determine the correct functioning of the contract, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine 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 changes between the following commits in the code provided by the developers in the following GitHub repository:

https://github.com/provable-things/ptokens-core-private

Start commit: `1e18c0a97dd9afd39b92b99709c737df94280a80`
Final commit: `b5a47d70afbbb9f5084efeb1c9807749e1395ad7` (tag `v1.11.0`)

This report presents the findings over 291 commits with 273 changed files, containing 9,786 additions and 9,637 deletions.

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

Project Overview

The submitted code implements a cross-blockchain bridge that allows assets to be moved between Ethereum and EOS as well as between Bitcoin and EOS. Bitcoin is represented on Ethereum and EOS as pBTC.

The two-way peg works by depositing (locking) BTC on Bitcoin and minting pBTC on Ethereum or EOS and by burning pBTC on Ethereum or EOS and unlocking BTC on Bitcoin. Transactions are relayed across chains through light clients designed to operate in a secure enclave.

The enclave is designed to be executed in a protected enclave using a trusted execution environment, such as Intel SGX.

Findings

1. EOS block submission does not enforce submission of blocks with schedule/producer set changes

Severity: Major

`submit_eos_block_to_core` in `core-private/src/btc_on_eos/eos/submit_eos_block.rs` does not enforce that it receives blocks with schedule changes. This allows blocks to be accepted from a block producer that has been removed from the schedule, opening the possibility of double-spending.

Recommendation

Fail if a block is submitted for a schedule that has not been confirmed by a block with schedule changes.

Status: Acknowledged

Due to lack of library support caused by moving schedule changes into `header_extenison` in EOS 2.0, the current version does not support parsing schedule changes. The issue described above is currently worked around by supplying schedule changes in a semi-trusted manner.

2. EOS block validation in `core-private` does not check whether block has been confirmed by enough producers

Severity: Medium

In the current implementation, `core-private` processes the supplied EOS block without any checks for its finality. This means that the core could be currently looking at a fork that might not end up in the canonical chain, leading to an inconsistent database or even to double-spending. While this issue is mitigated by the fact that the current implementation of `eos-syncer` filters actions by for irreversibility in `eos-syncer/lib/get-redeem-actions.js:36`, it adds a dependency on that component working as expected, requiring an additional trusted component in the system.

Recommendation

Only process irreversible blocks in `core-private`. Until realtime BFT has been added to EOS, the best approach is to track the finalized block (called last irreversible block or LIB in EOS) by ensuring that ⅔ + 1 block producers (15 out of the 21) have confirmed the current chain.

Status: Acknowledged

The current implementation of relying on `eos-syncer` to supply only irreversible blocks is intended for the alpha version of the bridge only. It is intended to submit blocks with relevant actions along with the subsequent blocks required for that relevant block to achieve irreversibility in order that the core knows that the block in question has reached finality.

3. Disabling validation when not in debug mode

Severity: Medium

It is possible to disable validation of blocks and their contents without explicitly requiring debug mode. This could lead to accidentally signing of txs in malicious blocks.

Recommendation

Require `DEBUG_MODE` to be explicitly switched on in order to deactivate `CORE_IS_VALIDATING`.

Status: Resolved

4. Linear memory increase of tx ids list

Severity: Minor

`maybe_add_tx_ids_to_processed_tx_ids` in `core-private/src/btc_on_eos/eos/add_tx_ids_to_processed_list.rs:38` does add more and more tx ids to the database over time, without ever pruning the state. This means that memory consumption of the enclave increases linearly over time, which could lead to the enclave crashing at some point.

Recommendation

Add state pruning to the processed tx ids list.

Status: Resolved

5. Nightly toolchain

Severity: Minor

The nightly toolchain is currently used. Secure applications should be developed with the fully stable toolchain.

Recommendation

Switch to the stable toolchain. Support for the `?` operator on `Option` has been added in Rust 1.22: https://doc.rust-lang.org/edition-guide/rust-2018/error-handling-and-panics/the-question-mark-operator-for-easier-error-handling.html

Status: Resolved

6. Code duplication

Severity: Minor

There is duplicated code in multiple places across the codebase, most notably between `core-private/src/btc_on_eos` and `core-private/src/btc_on_eth`. This duplication has been documented in `core-private/src/notes`. While this is not a vulnerability per se, it makes the codebase hard to maintain and increases the likelihood of bugs being introduced by changes that are not applied consistently across the duplicates. While some shared functionality has been factored out since the last audit report, there is still a lot of common behavior duplicated across those directories, which makes maintenance hard and can introduce regressions of bugs over time. Even worse, it could lead to separate implementations of the same functionality, which increases maintenance burden even more.

Recommendation

Extract shared behavior into shared crates/packages. Use a tool like duplo to find code duplication. De-duplication of the codebase does not just improve maintainability, it also speeds up the addition of bridges to further blockchains.

Status: Acknowledged

Refactoring of code has been continued since the last audit, but not finished before this audit report was released.

7. DB key collisions

Severity: Minor

Running multiple instances of the bridge on the same database can cause key clashes, which would result in information loss and corruption of the database.

Recommendation

Key collisions can be prevented by ensuring each bridge runs on a separate database, or by adding a distinct key prefix for each instance.

Status: Resolved

An environment variable `DB_KEY_PREFIX` has been added, which can be used to namespace keys for multiple bridge instances.

8. Bridge requires EOS names with more than 1 character

Severity: Minor

A condition in `core-private/src/btc_on_eos/btc/filter_too_short_names.rs:23` leads to EOS names with a length of one character not being valid recipients of tokens. Since EOS does indeed allow 1 character names, this would lead to valid token transfers not being executed properly.

Recommendation

Change the condition to require non-empty EOS names instead.

Status: Resolved

The condition was added as a workaround for a malicious transaction but is no longer required, so it was replaced with a non-empty check.

9. Output of debugging could accidentally trigger slashable behaviour

Severity: Minor

Since the core outputs data in the same format during debug mode as during production mode, a transaction sender service could accidentally submit transactions that were not intended to be sent. This could lead to lost funds, or worse, if offenses for double signing were added at some point, could cause slashing.

Recommendation

Prefix or otherwise modify the outputs of the core to invalidate them, e. g. by adding a "DEBUG" string to the beginning of a JSON. This would produce errors in components parsing the JSON, preventing accidental sending of txs.

Status: Resolved

Overall Code Quality

Since the last audit report, the already excellent code quality has been further improved. Best practice recommendations have largely been followed.