Cryptonics

Audit Report

pTokens Bridge pBTC on EOS

APRIL 24 2020

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 has been performed by

Philip Stanislaus and **Stefan Beyer**

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	Private keys can be read directly from the database	Medium	Resolved
4	Private keys are public on `EosPrivateKey` struct	Medium	Resolved
5	Private keys might remain in memory	Medium	Resolved
6	EOS block validation does not check whether producer is assigned to current slot	Minor	Resolved
7	DB keys/constants are the same in both `btc_on_eth` and `btc_on_eos`	Minor	Resolved
8	Linear memory increase of tx ids list	Minor	Acknowledged
9	Rust code can panic	Minor	Resolved
10	Nightly toolchain	Minor	Acknowledged
11	Code duplication	Minor	Acknowledged
12	Segwit not supported	Informational	Acknowledged
13	`eos-action-proof-maker` does not validate block header and individual actions	Informational	No issue
14	`get_latest_block_numbers` does not return latest EOS block number	Informational	Resolved
15	EOS block fetching is missing in `eos-and-btc-block-getter`	Informational	Resolved
16	No linter used in Rust codebase	Informational	No issue
17	Open TODOs in the codebase	Informational	No issue

This report contains 17 findings on 14 pages (plus one cover page).

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform an audit of the pTokens Bitcoin to EOS 2-way asset transfer bridge, forming part of the pTokens project (https://ptokens.io/).

The objectives of the audit are as follows:

1. Determine the correct functioning of the implementation in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine 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 the 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 authors of this report do not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The smart contract code has been provided by the developers in form of a compressed source code archive with the following SHA-256 hash:

```
pbtc-on-eos-for-auditors.zip
e3c7c6c71870da731b65969f1662afa0a1460382a87ab3fe3fd18540d9a8ee99
```

Subsequent fixes to the core where provided in a second compressed source code archive:

```
ptokens-core-for-audit.zip
ceac63aa74d0d0114a259402e6e8e19fd2b173d33ae1ead5b2ca91c22af46286
```

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 provided in 11 modules implements a cross blockchain bridge that allows assets to be moved between Bitcoin to EOS. Bitcoin is represented on EOS as pBTC.

The two-way peg works by depositing (locking) BTC on Bitcoin and minting pBTC on EOS and by burning pBTC on 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 it's 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. Private keys can be read directly from the database

Severity: Medium

With the exposed function `get_eos_private_key_from_db` in `core-private/src/btc_on_eos/eos/eos_database_utils.rs:211` private key can directly be read from the database.

Recommendation

Invert the control flow by passing the database as a parameter to an associated function `read_from_database` on the `EosPrivateKey`.

Status: Resolved

4. Private keys are public on `EosPrivateKey` struct

Severity: Medium

The `private_key` field is public on the `EosPrivateKey` struct in `core-private/src/btc_on_eos/eos/eos_crypto/eos_private_key.rs:34`.

Recommendation

Make the `private_key` field private. Instead of `put_eos_private_key_in_db`, defined in `core-private/src/btc_on_eos/eos/eos_database_utils.rs:201` use the `write_to_database` method on `PrivateKey` and delete the no longer needed `put_eos_private_key_in_db` function.

Status: Resolved

5. Private keys might remain in memory

Severity: Medium

The current implementation of the `std::ops::Drop` trait for `BtcPrivateKey` and `EthPrivateKey` in `core-private/src/btc_on_eos/btc/btc_crypto/btc_private_key.rs:127`, `core-private/src/btc_on_eth/btc/btc_crypto/btc_private_key.rs:127` and `core-private/src/btc_on_eth/eth/eth_crypto/eth_private_key.rs:92` could panic through calling `expect(...)`. A panic in Drop might lead to a missed drop. This could lead to exposure of sensitive data through memory that is not overwritten.

Recommendation

Match the `Result` and use a hard-coded value to zero the memory instead of panicking.

Status: Resolved

6. EOS block validation does not check whether producer is assigned to current slot

Severity: Minor

`validate_block_header_signature` in `core-private/src/btc_on_eos/eos/validate_signature.rs:138` currently checks whether the block producer is in the currently scheduled set of active producers, but it does not check whether the producer is assigned to produce a block at the current block height.

Recommendation

Follow EOS' algorithm to check whether the block producer is assigned to the current slot.

Status: Resolved

Added check here: `core-private/src/btc_on_eos/eos/validate_producer_slot.rs:46`

7. DB keys/constants are the same in both `btc_on_eth` and `btc_on_eos`

Severity: Minor

The DB keys/constants are the same in `core-private/src/btc_on_eos/btc/btc_constants.rs` and `core-private/src/btc_on_eth/btc/btc_constants.rs` as well as `core-private/src/btc_on_eos/utxo_manager/utxo_constants.rs` and `core-private/src/btc_on_eth/utxo_manager/utxo_constants.rs`, which could lead to problems when the same database is used for both enclaves.

Recommendation

Use unique constants across enclaves.

Status: Resolved

8. 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: Acknowledged

It is planned to mitigate this issue by implementing more efficient data structures for processed key tracking.

9. Rust code can panic

Severity: Minor

In one place, the Rust code can panic. It is generally preferred to use Results instead:

- `core-private/src/btc_on_eos/eos/eos_hash.rs:70`

Recommendation

Use an empty hash if parsing fails and add corresponding checks for emptiness where the type cast is used.

Status: Resolved

10. 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: Acknowledged

Automatic conversion between `Option` and `Result` is used across the codebase, which depends on the `try_trait`, a feature not currently in the stable toolchain.

11. 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.

Recommendation

Extract shared behaviour into shared crates/packages. Use a tool like duplo to find code duplication.

Status: Acknowledged

Refactoring of code has begun but not finished before the audit report was released.

12. Segwit not supported

Severity: Informational

Segwit is currently not supported, which is properly documented in code and documentation.

Status: Acknowledged

13. `eos-action-proof-maker` does not validate block header and individual actions

Severity: Informational

As stated in `eos-action-proof-maker/README.md`, `eos-action-proof-maker` does not currently validate the block header and individual actions.

Recommendation

Validating the block header and individual actions increases robustness by guaranteeing that data is not corrupted, but it does not increase the security of the tool. If a bad actor can tamper action data by modifying the `action_mroot`, they can likewise also modify the block's hash and the actions such that a validation step passes. This would just shift the trust required one level up.

Status: No issue

14. `get_latest_block_numbers` does not return latest EOS block number

Severity: Informational

`get_latest_block_numbers` in `core-private/src/btc_on_eos/get_latest_block_numbers/mod.rs:13` currently does not return the latest EOS block number.

Status: Resolved

15. EOS block fetching is missing in `eos-and-btc-block-getter`

Severity: Informational

`eos-and-btc-block-getter` does currently not contain functionality to get EOS blocks.

Recommendation

Add eos block fetching.

Status: Resolved

16. No linter used in Rust codebase

Severity: Informational

Currently, no linter is used in the Rust parts of the codebase.

Recommendation

Add a linter with a custom configuration that sticks to the maintainers preferences (e. g. line-width), see https://rust-lang.github.io/rustfmt/.

Status: No issue

17. Open TODOs in the codebase

Severity: Informational

There are TODOs in the codebase that should be resolved:

- `api-server/lib/get-info-route.js:38,44`
- `api-server/lib/utils.js:16`
- `app/README.md`
- `app/src/main.rs:18`
- `btc-syncer/README.md`
- `core-private/README.md`
- `core-private/src/lib.rs:13-15`
- `core-private/src/btc_on_eos/check_core_is_initialized.rs:53,72`
- `core-private/src/btc_on_eos/crypto_utils.rs:72`
- `core-private/src/btc_on_eos/utils.rs:25,35,84,103,178`

- `core-private/src/btc_on_eos/btc/filter_p2sh_deposit_txs.rs:155`
- `core-private/src/btc_on_eos/btc/parse_minting_params_from_p2sh_deposits.rs:22`
- `core-private/src/btc_on_eos/btc/parse_minting_params_from_p2sh_deposits.rs:121`
- `core-private/src/btc_on_eos/btc/parse_submission_material.rs:153`
- `core-private/src/btc_on_eos/btc/btc_test_utils/mod.rs:319`
- `core-private/src/btc_on_eos/debug_functions/mod.rs:1-2`
- `core-private/src/btc_on_eos/eos/filter_irrelevant_proofs.rs:47`
- `core-private/src/btc_on_eos/eos/parse_redeem_params.rs:23`
- `core-private/src/btc_on_eos/eos/parse_submission_material.rs:378`
- `core-private/src/btc_on_eth/check_enclave_is_initialized.rs:36`
- `core-private/src/btc_on_eth/btc/parse_minting_params_from_op_return_deposits.rs:427`
- `core-private/src/notes`
- `db-repl/README.md`
- `eos-action-proof-maker/README.md`
- `eos-action-proof-maker/src/eos_merkle_utils.rs:76`
- `eos-action-proof-maker/src/parse_eos_block.rs:16`
- `eos-action-proof-maker/src/types.rs:72,81,84`
- `eos-syncer/eos-syncer.js:94,95`
- `eos-syncer/lib/constants.js:8`
- `eos-syncer/lib/get-redeem-actions.js:9`
- `tx-broadcaster/README.md`

Recommendation

Resolve TODOs.

Status: No issue

Open tasks affect maintainability, not security.