tBTC Bitcoin Audit

Audit period	25-05-2020 / 31-05-2020
Auditor	Sergi Delgado Segura

1. Executive Summary

In May 2020, I was put in touch with Thesis to conduct a security assessment of tBTC, after the system was put on a 10-day emergency pause due to an unnoticed bug.

The main scope of the audit was to cover the Bitcoin related bits of tBTC, as well as the libraries it is depending upon (Summa bitcoin-spv and Summa relays). The audit started in May 25th 2020 and was conducted by me (Sergi Delgado Segura) over the course of 1.5 person-week.

1.1 Scope

The scope covered the **Bitcoin** code related to **Deposit** and **Redemption** located in the following repositories and provided commits:

keep-network/tbtc	100e5aae60aad209b0d3c6d495438d2b5b270fd3
summa-tx/bitcoin-spv	327f4f49c2e5a823406aa1e0ef384f30cfd8f768
summa-tx/relay	0031068c9a2851ddbefd07a00c1c93e22720955b

Given the limited timeframe, for the dependant libraries, the scope was set to prioritise the bits of code being directly used by tBTC.

Finally, keep-network/keep-ecdsa was also set in scope provided enough time was left to give it a peak once the rest of the code was checked.

1.2 Objectives

The objectives of the audit were to check how tBTC interacted with Bitcoin in the **Deposit** and **Redemption** phases. In order to do that, the first goal was set to understand the system as a whole, and the interactions that lead to the specific functionality. Once that was covered, special care was taken to check how Bitcoin transaction were crafted and parsed, given that what forced this audit to happen was precisely an error in this regard.

2. Issues

This section covers the issues found during the audit, including severity and suggested fixes. Issues are arranged by severity.

2.1 requestRedemption accepts invalid _redeemerOutputScripts

Severity

Critical

Description

During the redemption process, the redeemer provides an `outputScript` where to receive the initial deposit back (minus fees if applicable). Only **a valid script** of the four supported types (`P2PKH`, `P2SH`, `P2WPKH` and `P2WSH`) should be accepted. However, while the type of the provided script is checked (the script looks like a valid one), the actual content is not properly enforced. Therefore, script looking like valid, but being internally malformed will pass the validity checks. A redeemer providing this kind of output script will prevent the signers to succeed on the redemption process, and will eventually be able to claim the signers deposit. A malicious redeemer can make a profit by doing so, given that the signer is overcollateralized.

The main problem of this issue lies on the interaction between `requestRedemption` and BCUtils.extractHash. The latter is expecting sanitised inputs, while the former is not providing them.

Extended description

This issue affects both keep-network/tbtc and summa-tx/bitcoin-spv.

https://github.com/keep-network/tbtc/issues/658
https://github.com/summa-tx/bitcoin-spv/issues/161

Suggested fix

Either run sanity checks on the outputScript passed to the contract, or run them on BCUtils.extractHash.

2.2 `txOutputVector` expected to have less than 252 elements but not enforced

Severity

Major

Description

During the funding phase the depositor is requested to make a deposit to the signers Bitcoin address in order to end up receiving the corresponding amount of tbtc. The funding output index is restricted to be a `uint8` by the contract, meaning that any transaction funding the deposit from an output index higher than 255 cannot be proved. While this is a current restriction of the design, the actual limit is in `252 (0xFC)`, given the VARINT encoding of output index in transactions. This restriction is never enforced by the contract, meaning that a transaction funded by an output in the `253, 254 or 255` index will not be provable either, implying a loss of funds by the depositor.

This is specially relevant when the deposit comes from a wallet not directly controlled by the user, as may be the case of an exchange that batches withdraws in a transaction with multiple outputs.

Extended description

https://github.com/keep-network/tbtc/issues/647

Suggested fix

The ideal fix would be accepting outputs from any index. However, this is unfeasible given that output vectors are parsed by the contract so the correctness of the data can be verified (in terms of valid transaction structure), so parsing too many outputs would exceed the gas limit. Therefore, the only viable solution is to properly inform the user of the risks of entering the system using a wallet that they do not fully control, and discouraging so (the user should be aware that a transaction funding the system using an output over index X will imply direct loss of funds).

2.3 latestRedemptionFee incorrectly set after a fee bump.

Severity

Major

Description

Notice that this issue was discovered by ToB before me.

During the redemption process a signer can request a fee bump after `INCREASE_FEE_TIMER` as a claim that the transaction could not make it to a block with the current fee rate. After the first fee bump, the `latestRedemptionFee` is incorrectly set to the `_newOutputValue` meaning that, from that point on, a transaction moving a substantial amount of funds to fees (potentially all) will pass redemption checks. This hypothetical transaction will not be approved by the system to be signed though.

If all signers collude and also collude with a miner, they can claim the eth collateral back and swipe the deposit to fees, taking effectively all the funds back.

The severity of this issue may vary depending on if it is assessed by implications or likelihood. By running this attack the signers risk losing all the funds if the all-fee-transaction is seen by any party monitoring the calls to the contract and the Bitcoin blockchain, given that the all-fee-transaction is not an authorised data to be signed. Therefore, an attacker would risk loosing all the funds for a `TX_PROOF_DIFFICULTY_FACTOR` time period (6 blocks) for a ~66% profit on succeed.

Extended description

https://github.com/keep-network/tbtc/issues/652

Suggested fix

Set the properly on fee bumps.

2.4 PERMITTED_FEE_BUMPS is never used

Severity

Minor

Description

`PERMITTED_FEE_BUMPS` is supposed to limit the times the signers are allowed to bump the fee for a transaction. This is a dead code path left from previous versions of the code.

Extended description

https://github.com/keep-network/tbtc/issues/665

Suggested fix

Remove from the codebase.

2.5 Naming utxoSize

Severity

Minor

Description

The UTXO value is referred throughout the codebase as `utxoSize`. Given that a UTXO is a struct containing both a script and a value, calling it `utxoSize` can lead to misconceptions that may end up causing bugs.

Extended description

https://github.com/keep-network/tbtc/issues/660

Suggested fix

Rename `utxoSize` to `utxoValue`.

2.6 validateHeaderWork is never used (bitcoin-spv)

Severity

Minor

Description

`validateHeaderWork` is checking the the amount of work in a header is above the target. However, this function is never used, and it is replaced by the same functionality in other parts of the code.

Extended description

https://github.com/summa-tx/bitcoin-spv/issues/169

Suggested fix

Remove from the codebase.

3. Additional checks

Additional checks that may have not direct impact in the codebase.

3.1 Improve docs of bitcoin-spv contracts

Some docs of bitcoin-spv were outdated / wrongly set. This could have confuse devs while trying to use them leading to potential bugs. The spotted errors have been corrected and submitted as pull requests:

https://github.com/summa-tx/bitcoin-spv/pull/159

https://github.com/summa-tx/bitcoin-spv/pull/166

3.2 Signature forgery in keep-network/keep-ecdsa

keep-ecdsa has been checked against the usual suspects for signature forgery, given that tBTC accepts proof of fraud for signatures over non-authorised digests.

No apparent issue has been found in this regard.