bool-cmt-lockscript

Audit Report

Tue Jul 09 2024

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

bool-cmt-lockscript Audit Report

1 Executive Summary

1.1 Project Information

Description	The bool-cmt-lockscript is the main script responsible for deploying the BTC assets on the CKB network. The key components of this script include argument validation and operation authorization.
Type	Bridge
Auditors	ScaleBit
Timeline	Mon Jul 08 2024 - Tue Jul 09 2024
Languages	Rust
Platform	CKB
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/boolnetwork/bool-ckb-btc-bridge-v1
Commits	62c9767df8c236189d663d577aed403d5a642fb0 4baea9891e391733549831480ac8bbb87630e73c

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ERR	contracts/bool-cmt-lockscript/src/errors.rs	42bd19dcfbfcf53757e52c0ef282e7895575cfc1
MAI	contracts/bool-cmt-lockscript/src/main.rs	b6ef94fe944c9982eb2a4443a9c7f0a9f6b6a341
ENT	contracts/bool-cmt-lockscript/src/entry.rs	2d2cd133e165e1e35760a4c88f3cfd2cedbd5e09

1.3 Issue Statistic

Item	Count	Fixed	Partially Fixed	Acknowledged
Total	2	1	1	0
Informational	0	0	0	0
Minor	1	0	1	0
Medium	1	1	0	0
Major	0	0	0	0
Critical	0	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked CALL Return Values
- Functionality Checks
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Bool Network to identify any potential issues and vulnerabilities in the source code of the bool-cmt-lockscript smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

(1) Dependency Check

A comprehensive analysis of the software's dependency libraries was conducted using the Govulncheck tool.

(2) Automated Static Code Analysis

The code quality was examined using a code scanner.

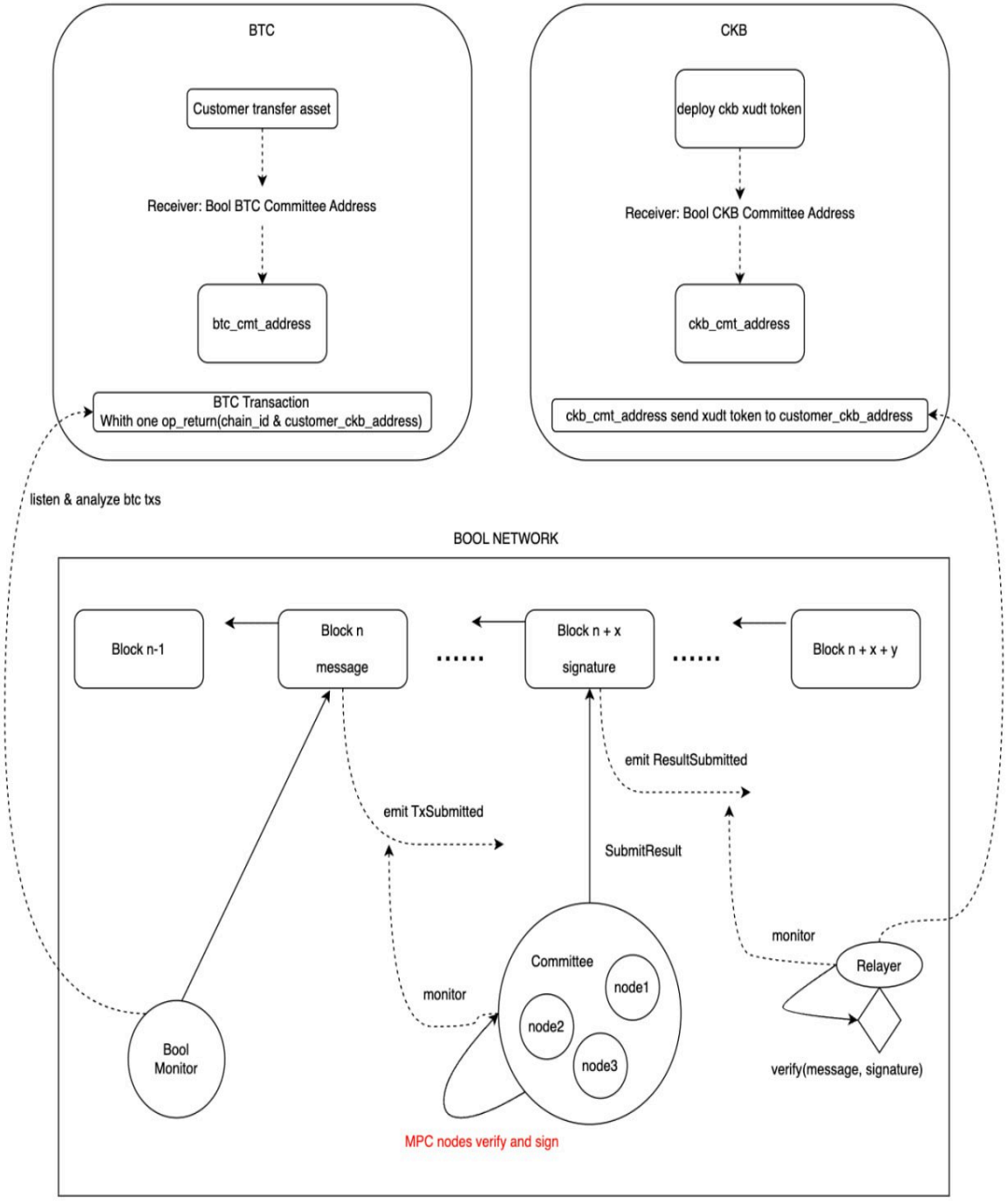
During the audit, we identified 2 issues of varying severity, listed below.

ID	Title	Severity	Status
ENT-1	Incomplete Checking of Inscription Formats	Medium	Fixed
ENT-2	Token Escrow Contracts Carry the Risk of Centralisation	Minor	Partially Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the bool-cmt-lockscript Smart Contract :
Take brc20 for example: BTC --> CKB: The user transfers the brc20 asset to BOOL's BTC committee address, and at the same time needs to carry an op_return output when constructing the transaction (which contains the corresponding chain_id of the CKB and the receiving address across the chain to the CKB), after the BOOL After network analysis and verification, BOOL's CKB committee address transfers the corresponding assets on the CKB network to the CKB address specified within the op_return in the BTC transaction. CKB --> BTC: The user transfers the brc20 asset on the CKB network to BOOL's CKB committee address, which needs to carry a cell_output to pay for the fee when constructing the transaction, as well as a witness containing the transaction signature and a cross-chain data (chain_id + token name + btc address), when BOOL listens to a transaction that matches the format of the transaction to verify the transaction, after the verification passes, BOOL's BTC committee address will transfer the corresponding assets on BTC to the btc address

specified in the witness.



4 Findings

ENT-1 Incomplete Checking of Inscription Formats

Severity: Medium

Status: Fixed

Code Location:

contracts/bool-cmt-lockscript/src/entry.rs#55-56

Descriptions:

The contract unlocks condition is incomplete for inscriptions, only checking for a length of 4 or 5 and not restricting the case. Bitcoin's inscriptions have ticks that can only be composed of lowercase letters.

```
fn is_valid_brc20(token: &str) -> bool {
    token.len().eq( other: &4) || token.len().eq( other: &5)
}
```

Suggestion:

Suggest adding a check that inputs can only be in lowercase.

Resolution:

In order to facilitate the adaptation of other scenarios in the future, this part of the logic is deleted, and the validation logic is transferred under the chain.

ENT-2 Token Escrow Contracts Carry the Risk of Centralisation

Severity: Minor

Status: Partially Fixed

Code Location:

contracts/bool-cmt-lockscript/src/entry.rs#30-43

Descriptions:

The unlocking condition of the contract is that the unlocking address is in the whitelist, which means that the whitelist of the contract needs to be fully trusted by the user, and there is a risk of centralisation at this point.

```
fn check_deploy_or_mint(script_hash: &[u8]) -> Result<(), Error> {
    let input_cell_lock_hashes : Vec<?> =
        QueryIter::new( query_fn: load_cell_lock_hash,  source: Source::Input).collect::<Vec<_>>();

    if input_cell_lock_hashes
        .into_iter()
        .any(|lock_hash| lock_hash[..] == script_hash[..])
        && !from_white_list()
    {
        return Err(Error::OperationNotAllowed);
    }

    Ok(())
}
```

Suggestion:

It is recommended that this whitelisted address be managed with multiple signatures or be handled by another decentralised solution.

Resolution:

This part of the code was modified so that anyone on the chain could post assets, but the Bool Network would only recognise transaction messages with a fixed multi-signature committee address. This mitigates the risk of centralisation to a certain extent.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.