DECENTER

Audit report for SpankChain

DECENTER

1. Summary

SpankChain has engaged Decenter in the period starting on October 25th and ending on October 30th 2018 to assess and audit their platform's Solidity smart contracts. This document describes the issues discovered during the audit. Decenter's assessment is focused primarily on code review of the contract, with an emphasis on security, gas usage optimization and overall code quality.

2. Audit

2.1 Authenticity

The audited contracts can be found in the GitHub repository: https://github.com/ConnextProject/contracts; the version used has commit hash *6720302e9051fb0fb824445ba7767874be67b873*.

2.2 Scope

This audit covered only the delivered *.sol files mentioned in the previous section.

2.3 Legend

🔴 High priority issues
🟠 Medium priority issues
🟡 Minor issues and optimisations
🟢 Notes and recommendations

3. Issues Found

---

1. Both hub and user can skip signature verification

File name: ChannelManager.sol
lines 175-237: hubAuthorizedUpdate() function,
lines 239-305: userAuthorizedUpdate() function

In case the hub sends the sigUser parameter (or the user sends sigHub parameter) as an empty string, the signature verification will be skipped, but channel balances will be updated.

This is a major issue and it allows one side to update the channel without the need for any verification from the other side.

This can be prevented by passing a boolean array indicator of length 2 to the _verifyThread() function where the verification of the hub's signature will only be performed if the first element of the array is true, and, likewise, the verification of the user's signature will only be performed if the 2nd element of the array is true.

[Amended] (31.10.2018)
This issue has been fixed in commit *f4162776fd0ab3f56fdea7596308c5098d1628b3*.

2. Wrong operator used for updating channel status

File name: ChannelManager.sol
lines 336-405: startExitWithUpdate() function,
lines 408-505: emptyChannelWithChallenge() function,
lines 508-592: emptyChannel() function

The operator == was mistakenly used in these three functions.

As a consequence, the channel status would not be updated as was intended.

This can be fixed simply by removing one of the equal signs (=), so that there is only one left.

[Amended] (31.10.2018)
This issue has been fixed in commit *c7602680ef1d0708471dd9d4803caf9cf90f4a78*.

3. Wrong signature passed to function when updating thread state

File name: ChannelManager.sol
lines 650-736: fastEmptyThread() function

The sig parameter is passed to the _verifyThread() function when verifying updated thread state.

As a consequence, this function cannot be executed successfully.

This can be remedied simply by using the updatedSig parameter instead.

[Amended] (31.10.2018)
This issue has been fixed in commit *35a37ffaaab4b13ad37e402fe25979ef6f752776*.

4. The hub cannot deposit funds to contract

File name: ChannelManager.sol;
Not included in the code.

Currently there is no function that will allow the hub to deposit weis or tokens.

Consequently, this means there is no simple way for the hub to acquire weis or tokens.

Adding a deposit function that will allow the hub exclusively to transfer weis and tokens to the contract would alleviate this issue and introduce a simple way for the hub to deposit weis or tokens.

5. Redundant noReentrancy modifiers increase gas usage

File name: ChannelManager.sol
lines 312-333: startExit() function,
lines 336-405: startExitWithUpdate() function,
lines 557-592: startExitThread() function,
lines 595-647: startExitThreadWithUpdate() function

This modifier is not needed in these functions because none of them make any external calls.

We recommend removing these modifiers, as in the current (Byzantine) fork of Ethereum this will reduce gas usage by ~10000. This will, however, change in the next (Constantinople) fork where the savings will only be ~200.

6. Redundant balance checks increase gas usage

File name: ChannelManager.sol
(lines 654-736): fastEmptyThread() function

The check for whether wei balance and token balance has been increased is included twice within this function. However, there is no need for the second check at lines 683 and 684, as there is no way for the wei balance or token balance to change between them and the previous checks.

We recommend removing the second check in order to reduce gas usage.

7. A redundant library is included

File name: Erecover.sol

The ERecover.sol library is included, but it is not used anywhere in the contracts.

We recommend removing the library in order to keep the codebase as clean as possible.

8. Typos in comments

FileName: ChannelManager.sol;
lines: 694; 707; 761; 774

The word “remaining” is misspelled as “remainining” in comments at lines 694, 707, 761 and 774.

9. Function should be moved to the lib folder

FileName: ChannelManager.sol;
lines 1054-1069

We recommend moving the _isContained function to the lib folder and turning it into a library, as it is in no way directly connected with the ChannelManager contract.

10. Different function would provide more transparency

FileName: ChannelManager.sol;
lines: 1015; 1019; 1047

We recommend using the isSignedBy() function at these lines as it includes information about who the signer is and would provide more transparency.

11. Examples in the comments are only about the user

FileName: ChannelManager.sol;
lines 892-931: _applyPendingUpdates() function;
lines 933-963: _revertPendingUpdates() function

Both examples contained in these lines contain information about the user, while one should actually be about the hub.

12. Older version of Solidity used

All .sol files.

An older version of Solidity is used. As of now the latest stable version is 0.4.25 and we recommend updating to and using the latest stable version.

4. Audit

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of business model, or any other statements about fitness of the contracts to purpose or their bugfree status. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

5. Closing Summary

It is the conclusion of this review that the codebase of the platform is well organized and appears to be sufficiently modular, even though the contract is highly complex. In general, the smart contract code adapts all the relevant best practices and has clean, legible code which will be further improved with updates to the issues mentioned in the audit. In addition to this, we believe that the documentation included in the source code could be improved. Please note that the main issues found by the audit are the items marked with high priority.