SOLIDIFIED

Audit Report for Golem (GNT Deposit Contract) on August 2nd, 2019.

Summary

Audit Report prepared by Solidified for Golem covering the GNT Deposit smart contract (and inherited dependencies).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on August 2nd, 2019, and the final results are presented here.

Audited Files

The following contracts were covered during the audit:

- GNTDeposit.sol
- ReceivingContract.sol

Notes

The audit was performed on commit `945ce6f42024ffbebac477783d827b32fdacb4c6` and pragma version 0.5.3

Intended Behavior

The contract allows tokens to be deposited/withdrawn by requesters and to be used to reimburse providers in the Golem ecosystem.

SOLIDIFIED

Audit Report for Golem (GNT Deposit Contract) on August 2nd, 2019.

Issues Found

Critical

No critical issues have been found.

Major

1. Signature Replay is Possible

While the subtask id is included in the signed message and this could act as a nonce, there is no registry of this to control whether the signatures have already been used nor for the value reimbursed per task. Since only the Concent address can invoke this, this may not be a serious issue depending on how the off-chain code is structured.

Recommendation
We recommend adding a nonce and keeping track of the amount refunded per task for this purpose, ensuring signatures are not reused and reimbursements do not surpass the value of the task. It is also considered a best practice to include the contract address in the signed data, to avoid signatures from another network (e.g. testnet) to be replayed.

Partially amended [28-08-2019]
Golem added the contract address as one of the inputs signed by the user, mitigating the risk of replay between different chains (Beware that it is possible to have the same or a different contract implemented in a different address in other networks/testnets, as long as it's deployed using the same keys and it falls in the same nonce for that specific key).

Golem opted not to address the risk of replay within the same chain, due to the fact that only the Concent can send these transactions, and it is controlled by Golem.

Golem's response:
`We have added the address of the contract to the input data which the signature is generated from to mitigate the testnet -> mainnet replay attack. As for the possibility of replay within a single chain, that possibility is eliminated in the Concent app itself and has been consulted with our legal team as an acceptable solution.`

2. Signature Verification does not Check for Malleable Addresses

The signature verification does not check for malleable addresses which are still allowed by `ecrecover`.

Recommendation
We recommend adding checks for malleable addresses. Here is an example of how to check for this:
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/cryptography/ECDSA.sol

Amended [28-08-2019]
The issue was amended and is no longer present in commit `915117a93807f35db856a4cbfc8f17b71c263f94`.

Minor

3. Modifying limits can lead to unexpected state

The contract owner is allowed to change the variables `maximum_deposits_total`, `maximum_deposit_amount`, `daily_reimbursement_limit` to any value, including ones that are lower than the current contract state.
There aren't any unintended consequences of this, but the contract does enter in a state that is assumed to never happen.

Golem's response [28-08-2019]
`We have decided that lowering the limits below the current contract state are not an issue -> the effect will be more or less the same as limits overflowing in a regular way and the contract will just need to wait until those values come back down below the new limits.`

Notes

4. Upgrade code to Solidity version > 0.5.3

The Solidity version the contracts are primarily written in is 0.5.3, a now known vulnerable version of Solidity.

Having a set and enforced compiler version is suggested. Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Recommendation
Refer to the Ethereum changelog (https://github.com/ethereum/solidity/blob/develop/Changelog.md) for the most up to date Solidity version. Contract elements may need to be updated in order to work with the latest Solidity version.

Amended [28-08-2019]
The issue was amended and is no longer present in commit `915117a93807f35db856a4cbfc8f17b71c263f94`.

5. Poorly configured parameters can lead to overflows

While unlikely, it is possible to overflow limits calculation if `maximum_deposit_amount` and `maximum_deposit_total` are configured with an extremely high value.

Recommendation

Consider using safemath for this and other arithmetic operations.

Amended [28-08-2019]

The issue was amended and is no longer present in commit `915117a93807f35db856a4cbfc8f17b71c263f94`.

6. Download OpenZeppelin through NPM

The contract uses an outdated version of OpenZeppelin contracts (the naming of the contracts show that the version used is pre 2.0) which were copied and edited manually, including an update to Solidity 0.5.3.

Recommendation

It is recommended to install OpenZeppelin through NPM and update it to the latest version, as the code is maintained, used and tested by a large community, ensuring bugs are found patched in a timely manner: https://github.com/OpenZeppelin/openzeppelin-contracts/issues?q=is%3Aissue+is%3Aclosed+label%3Abug

Golem's response [28-08-2019]

`All of our contracts are to be deployed once without any plans to upgrade them after that. Also GNTDeposit is supposed to work with already deployed contracts (GNT, GNTBatching) and it makes sense to keep these contracts (and their dependencies) untouched for consistency between testing and after deployment. Thus it’s a conscious decision to keep them as they are in the repository.`

SOLIDIFIED

Audit Report for Golem (GNT Deposit Contract) on August 2nd, 2019.

Closing Summary

Golem's GNT Deposit contract contains no critical issues. Two major issues have been identified, along with some notes for improvement.

We recommend the issues are amended, while the notes are up to Golem's discretion, as they mainly refer to improving the operation of the smart contract.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Golem platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.