SOLIDIFIED

Audit Report for Quanta on September 5, 2020

Summary

Audit Report prepared by Solidified for Quanta covering the Escrow system and QNTX token smart contracts (and their associated components).

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on August 21st 2020. Updates to issues encountered were submitted and verified. The final results are presented here.

Audited Files

The following contracts were covered during the audit:

https://gitlab.blockchainlabs.asia/quanta-v3/sc-qntx/tree/develop

Notes

The audit was based on commit `ebf8f8cf25ef043e2f38d7152987e89f4da3382b`, Solidity compiler version `0.6.0`.

Update: A version with fixes applied was supplied as commit number `7d2b73c0d701dce9a6cc849835726495a8ed533b`.

Intended Behavior

The smart contracts implement the QNTX token, an ERC20 token that additionally implements approveAndCall and Meta transactions through the Gas Station Network and two Escrow contracts.

SOLIDIFIED

Audit Report for Quanta on September 5, 2020

Executive Summary

Solidified found that the Quanta contracts contain four minor issues, in addition to three informational notes.

We recommend all issues are amended, while the notes are up to Quanta's discretion, as it refers to best practices.

Issues found:

Critical	Major	Minor	Notes
0	0	4	3

Issues Found

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1.Owner key holds lots of power

Given the contract architecture, the owner key can perform a myriad of vital functions, like transferring tokens out of escrows, changing the token itself, granting write access to EscrowData contract, to name a few.

If this particular key gets compromised it may critically affect the whole system.

Recommendation
Limit the owners actions in the smart contracts. Another possible solution is to use a multisig contract for the owner functionality.

2. `EscrowTimedRelease.sol` and `Escrwo3rdParty`: Contracts can become illiquid

Both escrow contracts allow the owner to move funds, or change the token, regardless if they're locked in behalf of a user or not. Therefore, It is possible for a user to try to withdraw his tokens, but the escrow contract does not have enough liquidity to fulfill it.

Recommendation
Only allow the owner to tokens that exceed the total amount of locked tokens from users.

3. TRI.sol: approveAndCall whitelist is only enforced for meta transactions

It's unclear if this behavior is intentional, but when `approveAndCall` is called through a meta transaction, a whitelist of destinations is enforced, but when called directly it allows the external call to be made to any contract.

Recommendation
Review the issue above, if not intended adjust `approveAndCall` to check for the target.

Update: The issue has been fixed in the latest version provided.

4. Return value of ERC20 transfer not verified.

In `Escrow3rdParty` and `EscrowTimedRelease` there are five ERC20 transfers for which the return value is not checked. In the case of QNTX there is no impact, since it reverts on failed transfers. However, this behaviour is not mandatory in the ERC20 standard. The source code layout seems to indicate potential reuse with other tokens. If a token that just returns false for failed transfers (i.e: BAT) is used with the contract, failed transfers will still result in successful transactions, including `release`, `sentBack` and `claim`.

Update: The issue has been fixed in the latest version provided.

Notes

5. Duplicated constant declarations

All escrow contract files declare the same list of constants. This code duplication may lead to inconsistencies. It is considered good practice to declare constants used by several files in a separate single source file.

Recommendation

Declare constants in a separate source file (e.g. `Constants.sol`) and import it in all the escrow contracts. It is also possible to refactor the code to have all escrow types inherit from `EscrowData`.
See Note 7 for related improvements.

Update: The issue has been fixed in the latest version provided.

6. Unnecessary Import of Safe Math

`EscrowTimedRelease.sol` and `Escrow3rdParty.sol` both import Openzeppelin's safe math library but also declare a custom `safeAdd()` function for 64 bit unsigned integers. The library is never used.

Recommendation
Remove unused imports.

Update: The issue has been fixed in the latest version provided.

7. Consider using Enums and Structs for improved readability

The `EscrowData` contract defines a few data structures in the form of constants and arrays, but it could be replaced with structs and enums to improve the readability of the contracts, without any drawbacks. Here's a proposed solution:

```
    enum Type {
       TimedEscrow, ThreePartyEscrow
    }

    enum State {
       Initiated, Completed, Cancelled
    }

    struct Escrow {
       Type type;
       State state;
       address sender;
```

```
        address receiver;
        address decider;
        uint256 amount;
        uint64 dateCreated;
        uint64 dateRealease;
        uint64 dateCompleted;
        uint64 dateCancelled;
    }
```

Update: The issue has been fixed in the latest version provided.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Quanta PLC 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.