SOLIDIFIED

Audit Report for Metislab - April 06, 2021

Summary

Audit Report prepared by Solidified covering the Metis Protocol vault smart contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on 5 April 2021, and the results are presented here.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/MetisProtocol/metis

Commit number: `1436c298a05732eb3cea787e28198799db1442d8`

The scope of the audit was limited to the following two files:

```
contracts
├── TokenVault.sol
└── ComVault.sol
```

Intended Behavior

The smart contracts implement two token vault contracts with time-limited claim and withdrawal functions.

SOLIDIFIED

Audit Report for Metislab - April 06, 2021

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	Medium	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Metislab - April 06, 2021

Issues Found

Solidified found that the Metis protocol contracts contain no critical issues, no major issues, 2 minor issues, in addition to 3 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	ComVault.sol and TokenVault.sol: No over/underflow protection	Minor	Resolved
2	TokenVault.sol: The owner can prevent users from claiming the Metis tokens	Minor	Resolved
3	ComVault.sol: ERC-20 return value ignored	Note	-
4	ComVault.sol: unbounded loop in claim() function	Note	-
5	TokenVault.sol: Unused variable	Note	-

SOLIDIFIED

Audit Report for Metislab - April 06, 2021

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. ComVault.sol and TokenVault.sol: No over/underflow protection

The contracts import the `safeMath` library and declare its use for `uint256` but do not make use of the library.
An overflow or underflow could not be exploited by a user, but it is possible for `TokenVault.sol` to malfunction due to incorrect values set by privileged roles.

Recommendation
Use the `safeMath` library.

Update
Resolved.

2. TokenVault.sol: The owner can prevent users from claiming the Metis tokens

The contract owner can prevent users from claiming the Metis tokens by setting the `_tge` to 0.
This implies that users funding the contract fully trust the owner.

Recommendation
Do not allow changing the `_tge` once it has been set (or set it in the constructor only).

Update

Resolved.

Informative Notes

3. ComVault.sol: ERC-20 return value ignored

The function `withdrawFund()` does not check the return value of the `transfer` call. Many ERC-20 implementations do not revert on failure but return false. This may lead to the call completing, even though the token transfer has failed.

Recommendation
Check the return type or use Open Zeppelin's `safeTransfer` wrapper.

4. ComVault.sol: unbounded loop in claim() function

Depending on the use case, if the list `arrangements_[msg.sender]` grows too large, the `claim` transactions will get more expensive and eventually will cost too much gas to complete, due to the block gas limit.

Recommendation
Ensure the contract is only used in settings where the number of arrangements per user is relatively small and does not grow too large over time. Alternatively, avoid unbounded iterations.

5. TokenVault.sol: Unused variable

The variable `uint256 _tge;` is only set in `setTge()` function and its value is never used.

Recommendation
Remove unused variables.

Audit Report for Metislab - April 06, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Metislab 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.