SOLIDIFIED

Audit Report for Hinet Ram on March 4th, 2020.

Summary

Audit Report prepared by Solidified for Hinet Ram covering the VanyToken smart contracts (and inherited dependencies).

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on March 4th, 2020, and the final results are presented here.

Audited Files

vanyContract.sol (and inherited contracts from OpenZeppelin).

Notes

The audit was performed on commit `2ccb8df43b17f8da1980cd788ce1b7c55286abd3` of repository https://github.com/Intoit82/VanyToken/. The audit was based on the Solidity compiler version `0.5.13`. Follow up was made on commit `398dc4a82945c95c42a727a8d0a8dece2a48842c`.

Intended Behavior

The contracts implement an ERC20 compliant token with fixed supply leveraging OpenZeppelin's ERC20 implementation (Version 2.5).

SOLIDIFIED

Audit Report for Hinet Ram on March 4th, 2020.

Issues Found

Critical

No critical issues found.

Major

No Major issues found.

Minor

No minor issues found.

Notes

1. `ERC20Capped` import is unnecessary

`ERC20Capped` is currently imported by VanyToken, but it could be omitted since without implementation of a Mint function the token is effectively capped.

Removing the import will prevent the contracts `ERC20Capped` inherits from being implemented, reducing the bytecode size to be deployed, with the added benefit of a cleaner ABI. The following contracts will no longer be part of the token if `ERC20Capped` is removed:

- `ERC20Capped`
- `ERC20Mintable`
- `MinterRole`

Recommendation
Remove the import an constructor of `ERC20Capped`.

Follow up [10.03.2020]
The issue was fixed and is no longer present in commit `398dc4a82945c95c42a727a8d0a8dece2a48842c`.

2. Consider using a recent compiler version, and locking pragma

Currently the contract can be compiled with any version of the Solidity compiler 0.5:

```
pragma solidity ^0.5.0;
```

Recommendation
Consider using a later version of the compiler (`0.5.13` was used in the audit) and locking pragma ensuring it is always compiled with the version it was developed and tested with.

Follow up [10.03.2020]
The issue was fixed and is no longer present in commit `398dc4a82945c95c42a727a8d0a8dece2a48842c`.

3. Consider using the keyword `ether` to improve readability of the initial supply

The line could be `_mint(msg.sender, 800000000 ether);`, making it easier to read and to reason about. Keep in mind the ether keyword will add 18 decimal places to the variable, so if the token's `decimals()` is modified to any other number this will also have to be updated accordingly (not the case with the current implementation that reads from `decimals()`, though in the current implementation the cap would have to be updated in case of decimal places change).

Additionally, consider making the cap/initial supply a constant. It makes the contract a bit safer for future development changes, without any compromises, since they don't use storage slots.

Follow up [10.03.2020]
The issue was fixed and is no longer present in commit `398dc4a82945c95c42a727a8d0a8dece2a48842c`.

SOLIDIFIED

Audit Report for Hinet Ram on March 4th, 2020.

Closing Summary

The contracts contain no security issues that could affect their behavior. The audit revealed some opportunities for improvement, here reported as notes. The notes present no security risk to the smart contracts, and mainly refer to readability improvements and minor gas savings, and though can be fixed at Hinet Ram's discretion.

The contracts were tested against known vulnerabilities such as overflows, reentrancy and others, as well as for ERC20 compliance and possible optimizations.

Follow up [10.03.2020]
All issues were fixed and are no longer present in commit
`398dc4a82945c95c42a727a8d0a8dece2a48842c`.

Disclaimer

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