MAINSTON

Security Assessment

September 22nd, 2020

By :
Camden Smallwood @ CertiK
camden.smallwood@certik.org

DISCLAIMER

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

SUMMARIES

Project Summary

Project Name	MAINSTON
Description	ERC-20 Token Smart Contract
Platform	Ethereum; Solidity
Codebase	GitLab Repository
Commits	master

Audit Summary

Delivery Date	Sep. 22, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Aug. 5, 2020 - Aug. 7 2020

Vulnerability Summary

Total Issues	3
Total Critical	0
Total Major	0
Total Minor	2
Total Informational	1

FINDINGS

ID	Title	Type	Severity
MAI-01	Incorrect implementation of function override	Implementation	Minor
MAI-02	Improper guarding against multiple calls to `removeUpgradeEngine`	Implementation	Minor
MAI-03	Incorrect spelling in comment	Grammar	Informational

MAI-01: Incorrect implementation of function override

Type	Severity	Location
Implementation	Minor	StonToken.sol L169-175

Description:

Due to C3 linearization, the right-most contract with the function in the override list is chosen, which is `ERC20Capped` in this case. This means that `super._beforeTokenTransfer` evaluates to `ERC20Capped._beforeTokenTransfer`, which is the original functionality without overriding the function, making the override unnecessary.

Recommendation:

We recommended that if the intention was to bypass the check created in `ERC20Capped`, specify the `ERC20` contract explicitly instead of using `super`.

Alleviation:

The recommendation was not taken into account. ValidityLabs conveyed that the intention was not to bypass the check created in `ERC20Capped`, so we suggest that the function be removed, as it only calls the implementation defined in the `ERC20Capped`, making it unnecessary to override the function at all.

MAI-02: Improper guarding against multiple calls to `removeUpgradeEngine`

Type	Severity	Location
Implementation	Minor	StonToken.sol L73-76

Description:

While this issue does not compromise the system, the owner of the token had the ability to call `removeUpgradeEngine` multiple times after minting due to the lack of requiring the swap engine address to be non-zero or setting an explicit `upgradeFinished` state.

Recommendation:

We recommended to either require the swap engine address to be non-zero, or add an `upgradeFinished` state and verify that it is set after removing the swap engine and added to the function's requirements.

Alleviation:

ValidityLabs conveyed that the issue was resolved and that the test files were updated accordingly.

MAI-03: Incorrect spelling in comment

Type	Severity	Location
Grammar	Informational	StonToken.sol L134

Description:

The `StonToken` contract had a misspelled comment:

```
  * @dev OVERRIDE method to forbide the possibility of renouncing ownership
```

Recommendation:

We recommended to change `forbide` to `forbid`.

Alleviation

ValidityLabs conveyed that the issue was resolved.