Security Assessment

Sota

May 26th, 2021

Table of Contents

Summary

This report has been prepared for Sota smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Sota
Description	SOTA token is a governane token of SOTA PLATFORM.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/sota-platform/sota-token-contracts/blob/master/contracts/HSotaToken.sol
Commits	68f37a7e408a0a851987bba4cfa44ea71e85274a

Audit Summary

Delivery Date	May 26, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	Openzeppelin ERC20

Vulnerability Summary

Total Issues	2
Critical	0
Major	0
Medium	0
Minor	1
Informational	1
Discussion	0

Audit Scope

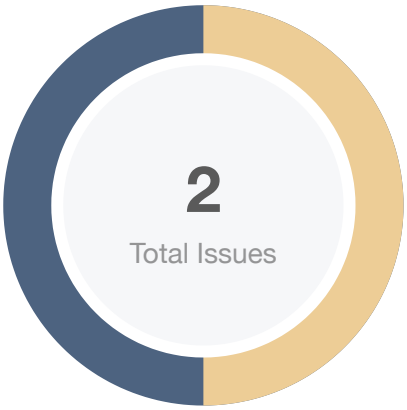
ID	file	SHA256 Checksum
HST	HSotaToken.sol	14ccb97978e4ef2d9679248937669a4b561a5886b08fb56699ab0839013da7d8

Review Summary

SOTA is a multi-chain digital content NFT P2P marketplace for creators and collectors.

The scope of the current audit is `HSotaToken.sol`, an `ERC20` token, a part of the whole protocol. It seems like a wise decision for the SOTA team to base the token on the `Openzeppelin ERC20` contract, preventing potential manipulations and attacks in the future.

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	0 (0.00%)
Minor	1 (50.00%)
Informational	1 (50.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
HST-01	Discussion on the usefulness of "_receiver"	Logical Issue	Informational	Resolved
HST-02	Do not set value zero after transfer the FEE	Logical Issue	Minor	Resolved

HST-01 | Discussion on the usefulness of "_receiver"

Category	Severity	Location	Status
Logical Issue	Informational	HSotaToken.sol: 41	Resolved

Description

What is the use of the parameter `_receiver`? Function `swap` burns `_amount` tokens from `msg.sender` address, but has not transferred or minted any tokens to the `_receiver` address. What is the intention of design on method `swap`?

Alleviation

[SOTA Team]: The _receiver address is use for emit event data, we have monitor system that watch the event data and get this address for our production features, no mint/transfer token to this address.

HST-02 | Do not set value zero after transfer the FEE

Category	Severity	Location	Status
Logical Issue	Minor	HSotaToken.sol: 55	Resolved

Description

Function `adminWithdrawFee` is only called by the owner, it allows the caller to withdraw the fee from this contract. The state variable `feeCollected` should be set zero after the operation. At the same time, it should be careful about the reentrancy attack.

Recommendation

We recommend changing the code such as the following example:

```
function adminWithdrawFee(address _to) public onlyOwner {
  uint256 currentFee = feeCollected;
  feeCollected = 0;
  _transfer(address(this), _to, currentFee);
}
```

Alleviation

SOTA team heeded our advise and changed the code. The recommendation was applied in the commit `a76d200dc6c8831e6ab4cf276b01a1845bf20cf9`.

Appendix

Finding Categories

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.