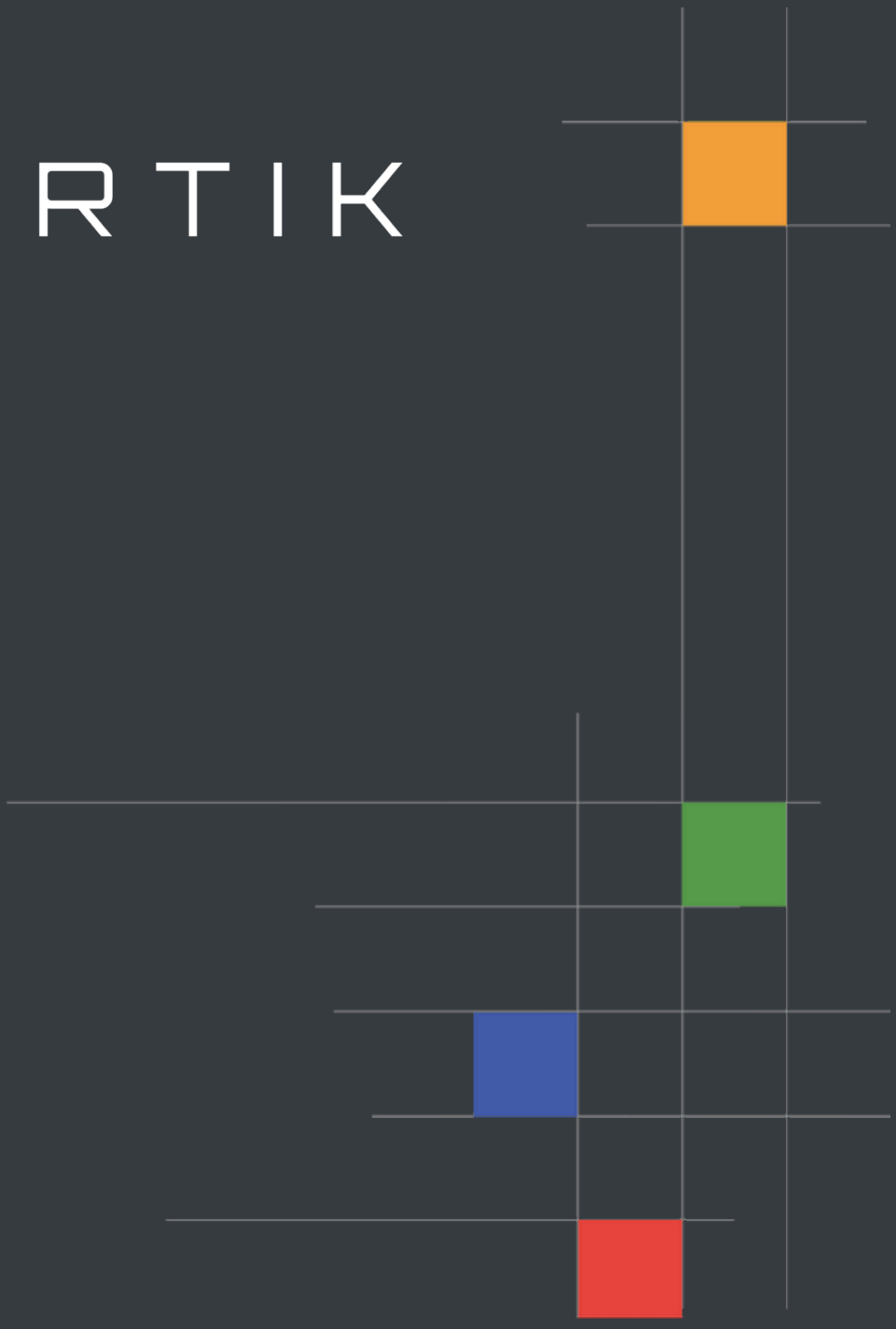
Singularity

DAO

Security Assessment

May 6th, 2021

Audited By:
Angelos Apostolidis @ CertiK
angelos.apostolidis@certik.org
Reviewed By:
Sheraz Arshad @ CertiK
sheraz.arshad@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 completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Singularity - DAO
Description	A typical governance token, based on the Compound Governance Token, where holders of this token have the ability to govern the protocol.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 7a7205be467982ea24598682d337da613a85a726 2. 88cba5a966705d9e14f7b415bf118b96789fb199

Audit Summary

Delivery Date	May 6th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	May 3rd, 2021 - May 6th, 2021

Vulnerability Summary

Total Issues	5
Total Critical	0
Total Major	0
Total Medium	1
Total Minor	1
Total Informational	3

Executive Summary

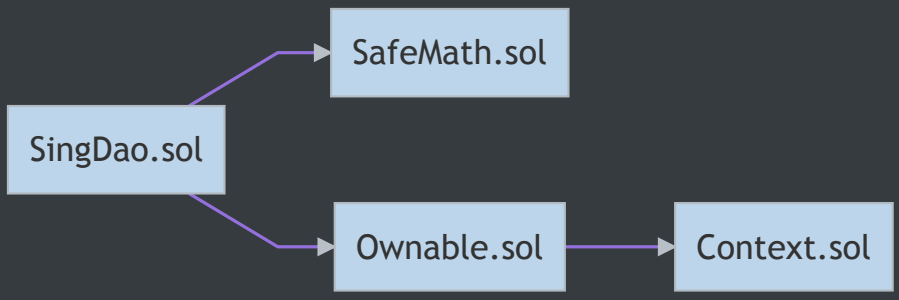
The report represents the results of CertiK's engagement with SingularityDAO on the implementation of their governance token smart contract.

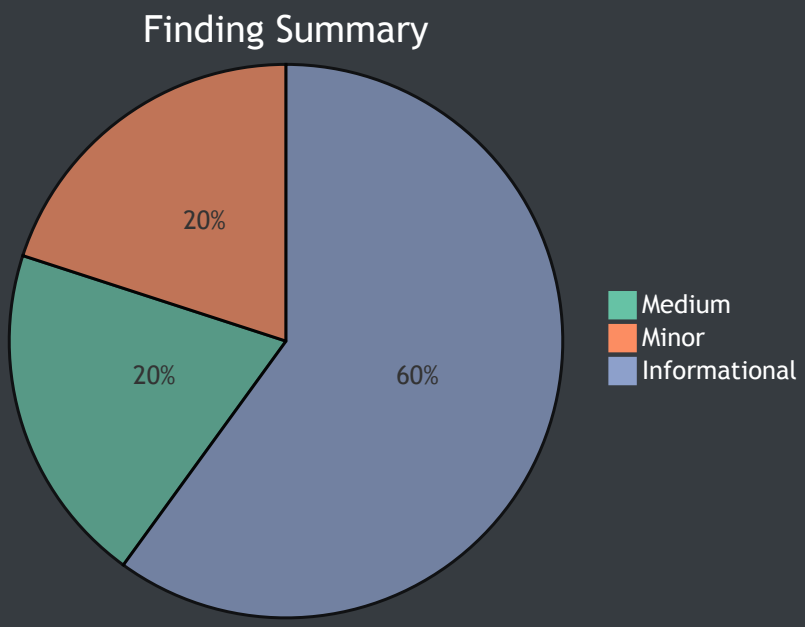
No notable vulnerabilities were identified in the codebase and it makes use of the latest security principles and style guidelines, even with the flexible minting mechanism that the contract owner can utilize. There were certain optimizations observed as well as security principles that can optionally be applied to the codebase to fortify the codebase to a greater extent.

Files In Scope

ID	Contract	Location
CON	Context.sol	Context.sol
OWN	Ownable.sol	Ownable.sol
SDO	SingDao.sol	SingDao.sol

File Dependency Graph





Manual Review Findings

ID	Title	Type	Severity	Resolved
OWN-01	Pull-Over-Push Pattern	Logical Issue	Minor	✓
SDO-01	Use of Mint Cap	Logical Issue	Medium	✓
SDO-02	Potential `locker`-less Contract	Logical Issue	Informational	
SDO-03	User-Defined Getters	Gas Optimization	Informational	
SDO-04	Inexistent Input Sanitization	Logical Issue	Informational	

OWN-01: Pull-Over-Push Pattern

Type	Severity	Location
Logical Issue	Minor	Ownable.sol L63-L67

Description:

The change of admin overrides the previously set admin with the new one without guaranteeing the new admin is able to actuate transactions on-chain.

Recommendation:

We advise the pull-over-push pattern to be applied here whereby a new owner is first proposed and consequently needs to accept the owner status ensuring that the account can actuate transactions on-chain.

Alleviation:

The development team opted to consider our references and used the pull-over-push pattern, as proposed, to manage the contract ownership.

SDO-01: Use of Mint Cap

Type	Severity	Location
Logical Issue	Medium	SingDao.sol L80-L92

Description:

The `mint()` function fails to check the newly minted amount against the mint cap.

Recommendation:

We advise to add a `require` statement, ensuring that the amount-to-mint will not exceed the mint cap set in the contract.

Alleviation:

The development team opted to consider our references and removed the minting cap per `mint()` invocation mechanism. We should note that the contract owner can still arbirtarily mint tokens and increase the total supply.

SDO-02: Potential `locker` -less Contract

Type	Severity	Location
Logical Issue	Informational	SingDao.sol L382-L384

Description:

Although the `setLocker()` function is only invocable by the contract `owner`, it fails to check the value of the new `locker` address.

Recommendation:

We advise to add a `require` statement, ensuring that the new `locker` is set as expected.

Alleviation:

The Singularity - DAO development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

SDO-03: User-Defined Getters

Type	Severity	Location
Gas Optimization	Informational	SingDao.sol L51, L122-L124

Description:

The linked variables contain user-defined getter functions that are equivalent to their name barring for an underscore (`_`) prefix / suffix.

Recommendation:

We advise that the linked variables are instead declared as `public` and that they are renamed to their respective getter's name as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation:

The Singularity - DAO development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

SDO-04: Inexistent Input Sanitization

Type	Severity	Location
Logical Issue	Informational	SingDao.sol L80-L92

Description:

Although the `mint()` function is only invocable by the contract `owner`, it fails to check the value of the `dst` address.

Recommendation:

We advise to add a `require` statement, checking the `dst` argument against the zero address.

Alleviation:

The Singularity - DAO development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

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