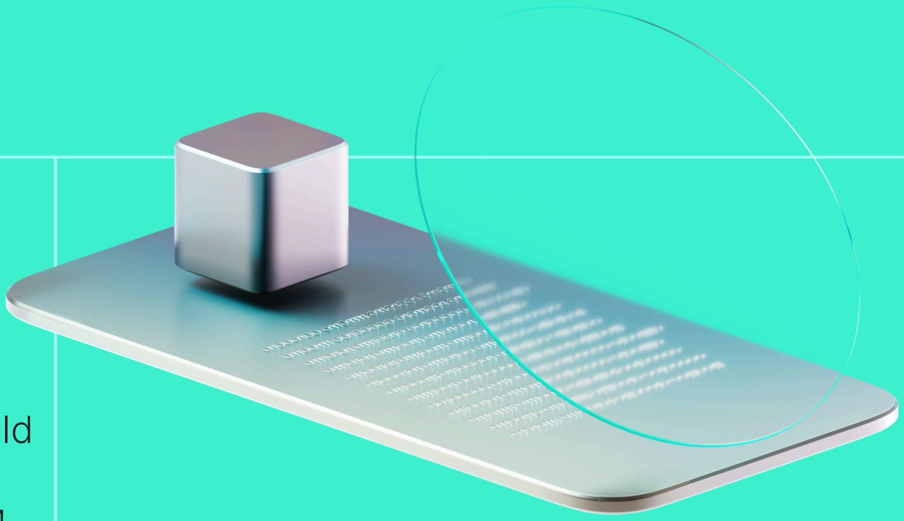
Smart Contract Code Review And Security Analysis Report



Customer: SerenityShield

Date: 29 February, 2024

We thank SerenityShield for allowing us to conduct a Smart Contract Security Assessment. This document outlines our methodology, limitations, and results of the security assessment.

Serenity Shield is innovative, secure, multi-chain data storage, transfer, recovery and inheritance solutions.

Platform: EVM

Language: Solidity

Tags: ERC-20

Timeline: 15.12.2023 - 15.12.2023

Methodology: Link

Last review scope

Repository	https://github.com/Decubate-com/smart-contracts/
Commit	6080661e3b5aae7b55dc8c6fb511c8571c0bdd9a
Deployed address	https://bscscan.com/token/0×84affEEf925Cdce87f8A99B7b2E540dA5140Fc09#code

View full scope

Audit Summary

10/10	10/10	0%	10/10
Security score	Code quality score	Test coverage	Documentation quality score

Total: 10/10

🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩

The system users should acknowledge all the risks summed up in the risks section of the report.

0	0	0	0
Total Findings	Resolved	Acknowledged	Mitigated

Findings by severity	Findings Number	Resolved	Mitigated	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	0	0	0	0

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for SerenityShield
Approved By	Yves Toiser - SC Lead Auditor at Hacken OÜ
Audited By	Viktor Raboshchuk - SC Auditor at Hacken OÜ
Website	https://www.serenityshield.io/
Changelog	15.12.2023 – Preliminary Report

Introduction

Hacken OÜ (Consultant) was contracted by SerenityShield (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

System Overview

SerenityShield is a dApp protocol which contains ERC20 tokens for the interaction with the ecosystem, this audit covers the following contract:

- SERSH token — A simple ERC-20 token that mints all the initial supply to a deployer. Additional minting is not allowed. A burnable feature is implemented.

  It has the following attributes:

  - Name: SerenityShield
  - Symbol: SERSH
  - Decimals: 18
  - Total supply: 100 millions tokens.

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.
- NatSpecs are provided.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.
- There are no code quality issues.

Test coverage

Code coverage of the project is **0%** (branch coverage):

- Tests are not provided (according to our methodology, tests are not mandatory for projects smaller than 250 lines of codes).

Security score

As a result of the audit, the code contains **0** issues. The security score is **10** out of **10**.

Summary

According to the assessment, the Customer's smart contract has the following score: **10**. The system users should acknowledge all the risks summed up in the risks section of the report.

Risks

- All the tokens are minted to a single address. The secureness of the supply depends on the secureness of key storage.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

No medium severity issues were found.

■ Low

No low severity issues were found.

Informational

No informational issues were found.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Risk Levels

Critical: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

High: High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.

Medium: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

Low: Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope details

Repository	https://github.com/Decubate-com/smart-contracts/
Commit	6080661e3b5aae7b55dc8c6fb511c8571c0bdd9a
Deployed address	https://bscscan.com/token/0×84affEEf925Cdce87f8A99B7b2E540dA5140Fc09#code
Whitepaper	https://assets-global.website-files.com/646e1d2bb86f8833c75bdc7b/655e43d973850f9cf58b1860_WP%20SERENITY%20SHIELD%20(En)-min.pdf
Requirements	-
Technical Requirements	-

Contracts in Scope

./contracts/SERSHToken.sol

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

This document is proprietary and confidential. No part of this document may be disclosed in any manner to A third party without the prior written consent of Hacken.

https://hacken.io/