CODESPECT

REDSTONE TOKEN

SECURITY ASSESSMENT REPORT

4 November, 2024

Prepared for

RedStone

Contents

1 About CODESPECT

CODESPECT is a specialized smart contract security firm dedicated to ensuring the safety, reliability, and success of blockchain projects. Our services include comprehensive smart contract audits, secure design and architecture consultancy, and smart contract development across leading blockchain platforms such as Ethereum (Solidity), Starknet (Cairo), and Solana (Rust).

At CODESPECT, we are committed to building secure, resilient blockchain infrastructures. We provide strategic guidance and technical expertise, working closely with our partners from concept development through deployment. Our team consists of blockchain security experts and seasoned engineers who apply the latest auditing and security methodologies to help prevent exploits and vulnerabilities in your smart contracts.

Smart Contract Auditing: Security is at the core of everything we do at CODESPECT. Our auditors conduct thorough security assessments of smart contracts written in Solidity, Cairo, and Rust, ensuring that they function as intended without vulnerabilities. We specialize in providing tailored security solutions for projects on EVM-compatible chains and Starknet. Our audit process is highly collaborative, keeping clients involved every step of the way to ensure transparency and security. Our team is also dedicated to cutting-edge research, ensuring that we stay ahead of emerging threats.

Secure Design & Architecture Consultancy: At CODESPECT, we believe that secure development begins at the design phase. Our consultancy services offer deep insights into secure smart contract architecture and blockchain system design, helping you build robust, secure, and scalable decentralized applications. Whether you're working with Ethereum, Starknet, or other blockchain platforms, our team helps you navigate the complexity of blockchain development with confidence.

Tailored Cybersecurity Solutions: CODESPECT offers specialized cybersecurity solutions designed to minimize risks associated with traditional attack vectors, such as phishing, social engineering, and Web2 vulnerabilities. Our solutions are crafted to address the unique security needs of blockchain-based applications, reducing exposure to attacks and ensuring that all aspects of the system are fortified.

With a focus on the intersection of security and innovation, CODESPECT strives to be a trusted partner for blockchain projects at every stage of development and for each aspect of security.

2 Disclaimer

Limitations of this Audit: This report is based solely on the materials and documentation provided to CODESPECT for the specific purpose of conducting the security review outlined in the Summary of Audit and Files. The findings presented in this report may not be comprehensive and may not identify all possible vulnerabilities. CODESPECT provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, is entirely at your own risk.

Inherent Risks of Blockchain Technology: Blockchain technology is still evolving and is inherently subject to unknown risks and vulnerabilities. This review focuses exclusively on the smart contract code provided and does not cover the compiler layer, underlying programming language elements beyond the reviewed code, or any other potential security risks that may exist outside of the code itself.

Purpose and Reliance of this Report: This report should not be viewed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. Third parties should not rely on this report for any purpose, including making decisions related to investments or purchases.

Liability Disclaimer: To the maximum extent permitted by law, CODESPECT disclaims all liability for the contents of this report and any related services or products that arise from your use of it. This includes but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services: CODESPECT does not warrant, endorse, or assume responsibility for any third-party products or services mentioned in this report, including any open-source or third-party software, code, libraries, materials, or information that may be linked to, referenced by, or accessible through this report. CODESPECT is not responsible for monitoring any transactions between you and third-party providers. We strongly recommend conducting thorough due diligence and exercising caution when engaging with third-party products or services, just as you would for any other product or service transaction.

Further Recommendations: We advise clients to schedule a re-audit after any significant changes to the codebase to ensure ongoing security and reduce the risk of newly introduced vulnerabilities. Additionally, we recommend implementing a bug bounty program to incentivize external developers and security researchers to identify and disclose potential vulnerabilities safely and responsibly.

Disclaimer of Advice: FOR AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, AND ANY ASSOCIATED SERVICES OR MATERIALS SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Risk Classification

Severity Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Table 1: Risk Classification Matrix based on Likelihood and Impact

3.1 Impact

- **High** - Results in a substantial loss of assets (more than 10%) within the protocol or causes significant disruption to the majority of users.
- **Medium** - Losses affect less than 10% globally or impact only a portion of users, but are still considered unacceptable.
- **Low** - Losses may be inconvenient but are manageable, typically involving issues like griefing attacks that can be easily resolved or minor inefficiencies such as gas costs.

3.2 Likelihood

- **High** - Very likely to occur, either easy to exploit or difficult but highly incentivized.
- **Medium** - Likely only under certain conditions or moderately incentivized.
- **Low** - Unlikely unless specific conditions are met, or there is little-to-no incentive for exploitation.

3.3 Action Required for Severity Levels

- **Critical** - Must be addressed immediately if already deployed.
- **High** - Must be resolved before deployment (or urgently if already deployed).
- **Medium** - It is recommended to fix.
- **Low** - Can be fixed if desired but is not crucial.

In addition to High, Medium, and Low severity levels, CODESPECT utilizes two other categories for findings: **Informational** and **Best Practices**.

a) **Informational** findings do not pose a direct security risk but provide useful information the audit team wants to communicate formally.

b) **Best Practices** findings indicate that certain portions of the code deviate from established smart contract development standards.

4 Executive Summary

This document presents the security assessment conducted by CODESPECT for the smart contracts of Redstone. Redstone is an oracle protocol that provides customizable and cost-efficient data streams. Unlike traditional oracle providers, Redstone employs a unique approach where data is distributed by nodes to a decentralized data layer. These data are signed and utilized across various blockchain applications. The primary on-chain component of Redstone oracle is responsible for verifying data to ensure its reliability and authenticity, confirming that the data originates from trusted nodes.

This audit focuses on Redstone's ERC20 token, named `RedstoneToken`. This token inherits from the ERC20 standard and introduces additional minting functionality, allowing a privileged user, known as a minter, to mint tokens for any user. The token's maximum supply is enforced at the code level, capping it at fifty million tokens.

The audit was performed using:

a) Manual analysis of the codebase.

b) Simulation of the smart contract.

c) Creation of test cases.

We report 1 point of attention, classified as `Best Practices`. An overview of the issues is provided in Fig. 1.

Organization of the document is as follows:

- **Section 5** summarizes the audit.
- **Section 6** presents the issues.
- **Section 7** contains additional notes for the audit.
- **Section 8** discusses the documentation provided by the client for this audit.
- **Section 9** presents the compilation and tests.

Issues found:

Severity	**Count**	**Fixed**	**Acknowledged**
Critical			
High			
Medium			
Low			
Best Practices	1	1	0
Informational			
Total	**1**	**1**	**0**

Table 2: Summary of Issues Found, Fixed, and Acknowledged

5 Summary of the Audit

Audit Type	Security Review
Project Name	Redstone
Type of Project	ERC20
Duration of Engagement	1 Week
Duration of Fix Review Phase	1 Day
Draft Report	Sep 30, 2024
Final Report	Nov 4, 2024
Repository	redstone-oracles-monorepo
Commit (Audit)	ff0f3dcb085f28bd80ddc096825701db6e14d0af
Commit (Final)	20dd743aa3dbb1374278ce17e27ae391815e4ef4
Documentation Assessment	Not applicable here
Test Suite Assessment	High
Auditors	Talfao and Bloqarl

Table 3: Summary of the Audit

5.1 Scope - Audited files

	Contract	LoC	Comments	Ratio	Blank	Total
1	packages/eth-contracts/contracts/RedstoneToken.sol	21	5	23.8%	8	34
	Total	**21**	**5**	**23.8%**	**8**	**34**

5.2 Summary of Issues

	Finding	Severity	Status
1	Use a two-step mechanism for `minter` role transfer	Best Practices	Fixed

6 Issues

6.1 [Best Practices] Use a two-step mechanism for `minter` role transfer

File(s): `RedstoneToken.sol`

Description: The `RedstoneToken` contract extends the ERC20 standard by adding minting functionality. This allows the designated `minter` to mint any amount of tokens (provided it does not exceed the maximum supply) to any user through the `mint(...)` function.

The `minter` role can be updated using the `updateMinter(...)` function:

```
function updateMinter(address newMinter) external {
    require(msg.sender == minter, "RedstoneToken: minter update by an unauthorized address");
    minter = newMinter;
    emit MinterUpdate(newMinter);
}
```

This function directly updates the `minter` to the new address. However, this single-step mechanism is susceptible to errors and potential misuse. Implementing a two-step mechanism, as demonstrated in the `Ownable2Step` library from OpenZeppelin, would provide a safer and more controlled process for transferring the `minter` role.

Recommendation(s): Consider changing the single-step minter role transfer mechanism to a two-step process.

Status: Fixed

Resolved in: 20dd743aa3dbb1374278ce17e27ae391815e4ef4

7 Additional Notes

This section provides supplementary auditor observations regarding the code. These points were not identified as individual issues but serve as informative recommendations to enhance the overall quality and maintainability of the codebase.

- Lack of zero address checks in the `updateMinter(...)` and `mint(...)` functions.
- The check inside the `mint(...)` function to ensure that the new `total_supply` does not exceed the max supply should be enforced before the `_mint(...)` call to revert earlier.
- The max supply check is not enforced on the `constructor` level.

```
require(totalSupply() <= MAX_SUPPLY, "RedstoneToken: cannot mint more than MAX SUPPLY");
```

- The `event MinterUpdate(...)` could also contain the previous ‘minter‘ for more informational events.

8 Evaluation of Provided Documentation

Comprehensive documentation is essential to ensure clarity and transparency in smart contract development. It serves as a foundation for understanding the contract's functionality, architecture, and intended usage, providing valuable insights for developers, users, and stakeholders alike. Effective documentation also streamlines future maintenance by enabling efficient updates and modifications.

Several key types of documentation are commonly used in smart contract projects:

- **Technical Whitepaper**: Outlines the smart contract's purpose, architectural design, components, and interactions, giving a high-level overview of the system.
- **User Guide**: Provides instructions on interacting with the contract, with detailed steps for various functions and a summary of its features and use cases.
- **Code Documentation**: Offers an in-depth look into the codebase, including explanations of functions, variables, and components, as well as details on their specific roles within the contract.
- **Testing Documentation**: Contains information on test coverage, specific test cases, and results, providing insights into the contract's reliability and identifying any areas needing improvement.

Each type of documentation plays a critical role in ensuring that the smart contract is robustly designed, clearly explained, and thoroughly verified, thereby facilitating long-term reliability and ease of maintenance.

9 Test Suite Evaluation

9.1 Compilation Output

```
> yarn compile
Generating typings for: 13 artifacts in dir: typechain-types for target: ethers-v5
Successfully generated 44 typings!
Compiled 13 Solidity files successfully (evm target: london).
```

9.2 Tests Output

```
> yarn test

RedStone token
     Should properly transfer tokens
     Should not mint from account other than the minter
     Should mint from minter account
     Should mint max supply
     Should not mint more than max supply
     Should not update minter from unauthorized account
     Should update minter
```