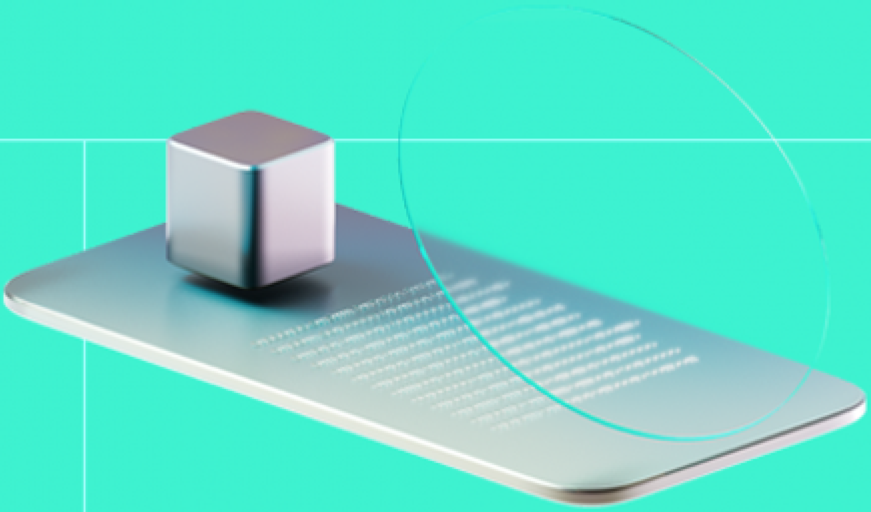
Smart Contract Code Review And Security Analysis Report

Customer: SatoshiSync

Date: 06/03/2024



We express our gratitude to the SatoshiSync team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Satoshisync is a permissionless chain agnostic protocol for BTC L2 aiming to facilitate seamless bridging of BRC20 assets between their native chain and any EVM compatible chain.

Platform: EVM

Language: Solidity

Tags: Bridging, BRC20, Ordinals, BTCFi

Timeline: 29/02/2024 - 01/03/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/
Commit	0bec464e

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

1	1	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	1
High	0
Medium	0
Low	0

Vulnerability	Status
F-2024-1190 - mint() function will not mint any wrapped assets, resulting in the loss of all tokens intended to bridge	Fixed

Document

Name	Smart Contract Code Review and Security Analysis Report for SatoshiSync
Audited By	Philipp Eder
Approved By	Yves Toiser
Website	https://hacken.io
Changelog	04/03/2024 - Preliminary Report & 06/03/2024 - Final Report

Table of Contents

System Overview

SatoshiSync is a permissionless, chain agnostic protocol for inscriptions and BTC L2.

It aims to facilitate easy customization and one-click bridging of BRC20 assets between their native chain and any EVM compatible chain.

WrappedBRC20.sol — a contract to mint wrapped assets on any EVM compatible chain as a result of bridging the BRC20 assets.

Privileged roles

- The owner of the smart contract has the sole privilege to mint wrapped assets.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

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

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

Code quality

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

Test coverage

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

Security score

Upon auditing, the code was found to contain **1** critical, **0** high, **0** medium, and **0** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **10**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The code is intended to be deployed within a Beacon Proxy pattern however, the framework to facilitate this structure was not provided.
- The bridging functionality, particularly the procedure to invoke the mint() function for generating wrapped assets lacks any kind of documentation, leaving this critical process unaddressed and unexplained.
- The security of the bridging process is reliant on the security of the third-party bridging mechanism, which is not subject of examination within the scope of this audit.
- The bridging mechanism depends on the contract owner's ability to mint tokens, making it essential for the bridging mechanism to hold the owner role for effective operation.
- The smart contract appears to offer upgradeability features, yet remains incomplete. Although the owners assert their intention to render it non-upgradeable, their capacity to enable upgrades presents a significant risk, demanding implicit trust in their commitment to finalizing the contract's non-upgradeable state.

Findings

Vulnerability Details

F-2024-1190 - mint() function will not mint any wrapped assets, resulting in the loss of all tokens intended to bridge - Critical

Description: The conditional statement `if (amount == 0) _mint(to, amount);` within the mint() function makes it impossible to mint wrapped tokens. Therefore any attempt to bridge funds will result in a complete loss of the tokens sent.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Classification

Severity: Critical

Impact:

```
Likelihood [1-5]: 5
Impact [1-5]: 5
Exploitability [1-2]: 0
Complexity [0-2]: 0
Final Score: 5.0 [CRITICAL]
```

Recommendations

Recommendation: change the conditional statement in order for `mint()` to be able to mint wrapped tokens.

Remediation: The client has fixed this vulnerability.

Observation Details

F-2024-1163 - Missing event emission - Info

Description:

The contract WrappedBRC20.sol lacks events to track important operations like minting or rescuing native tokens.

Events in smart contracts are essential for tracking changes on the blockchain, especially for key administrative actions.

Without events, tracking changes becomes challenging, reducing transparency and making it harder to verify actions retrospectively. This absence hinders external systems and interfaces from efficiently monitoring and reacting to important state changes in the contract

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Introduce specific events for functions to log significant activities:

- For `rescueNativeToken()`, emit an event capturing both the beneficiary's address and amount.

Remediation: The client has fixed this observation.

F-2024-1184 - Redundant payable flag in initializer function - Info

Description: The initialize() function within the WrappedBRC20.sol smart contract is marked as payable without any apparent reason.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Remove the payable flag from WrappedBRC20.initialize() function.

Remediation: The client has fixed this observation.

F-2024-1185 - Solidity style guide violation - Info

Description: The contracts WrappedBRC20.sol & CheckerZeroAddr.sol violate the solidity style guide by arbitrary placement of functions (regarding their visibility) & state variable declarations.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Inside each contract, library or interface, use the following order:

Type declarations
State variables
Events
Errors
Modifiers
Functions

Functions should be grouped according to their visibility and ordered:

constructor
receive function (if exists)
fallback function (if exists)
external
public
internal
private

Remediation: The client has fixed this observation.

F-2024-1192 - Unused function in CheckerZeroAddr.sol - Info

Description: The function `function __CheckerZeroAddr_init_unchained()` is present in CheckerZeroAddr.sol but never used.

Unused functions unnecessarily increase deployment cost.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Remove the unused function.

Remediation: The client has fixed this observation.

F-2024-1193 - Unused interface with functions differing from implementation contract - Info

Description:	The interface ITokensRescuer.sol and its intended implementation contract TokensRescuer.sol are in no way connected through inheritance, furthermore the interface contains multiple functions not present in the implementation.
Assets:	• contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]
Status:	Fixed

Recommendations

Recommendation:	Make the contract TokensReceiver.sol inherit from the Interface ITokensReceiver.sol and implement either all functions in both or delete unused functions or remove the interface ITokensReceiver.sol **Remediation:** The client has fixed this observation.

F-2024-1213 - Functions not used internally can be marked as external - Info

Description: The mint() function is currently set to public visibility but never called internally.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Reduce the function visibility of mint() to external.

Remediation: The client has fixed this observation.

F-2024-1214 - Missing handling of negative case in conditional - Info

Description: The function mint() with the body `if (amount == 0) _mint(to, amount);` is missing the handling of a negative case.

Assets:

- contracts/core/WrappedBRC20.sol [https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/-/blob/main/contracts/core/WrappedBRC20.sol]

Status: Fixed

Recommendations

Recommendation: Add a require statement and revert string or a custom error to handle negative cases of the conditional.

Remediation: The client has fixed this observation.

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 **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
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.

Appendix 2. Scope

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

Scope Details

Repository	https://gitlab.com/hacken-audit-contracts/satoshisync-wrappedbrc20/
Commit	0bec464e73797953e143236f1b14eae0293a7fcf
Whitepaper	https://satoshisync.com/lightpaper.pdf
Requirements	https://syncsatoshi.gitbook.io/welcome/
Technical Requirements	Documentation.docx.pdf

Contracts in Scope

./contracts/core/WrappedBRC20.sol

./contracts/extensions/CheckerZeroAddr.sol

./contracts/extensions/TokensRescuer.sol