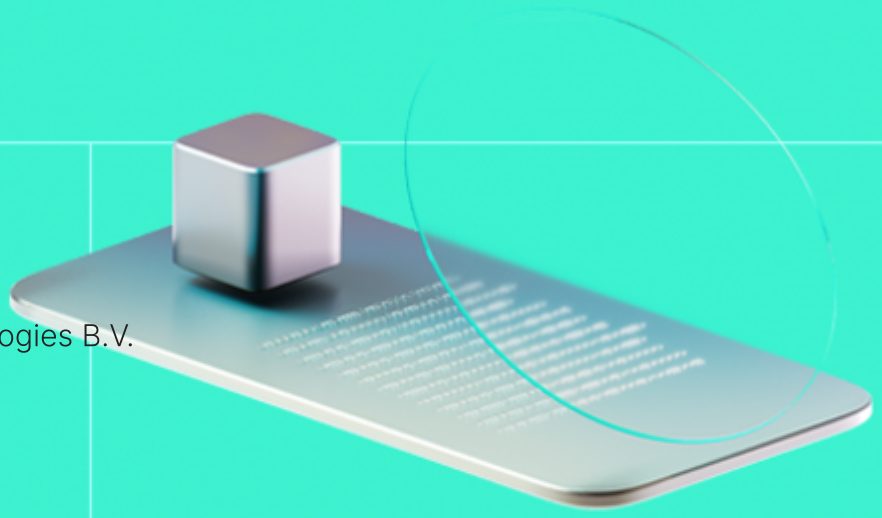
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Bloqhouse Technologies B.V.

Date: 24/03/2023

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

Bloqhouse Technologies offers a real world asset tokenization protocol, which aims to enable asset providers to fractionalize assets such as, but not limited to, real estate, land, property, on the blockchain as NFTs.

Platform: EVM

Language: Solidity

Tags: ERC721; RWA

Timeline: 06/03/2023 - 24/03/2023

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://bitbucket.org/alfredpersson/token-shares-solidity/src/master
Commit	cbdc7c0d6162346b96cf62cb2ff93c15f416819e

Audit Summary

10/10	10/10	100%	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

7	7	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	4
Medium	0
Low	3

Vulnerability	Status
F-2023-0606 - EIP Standard Violation	Fixed
F-2023-0607 - Data Consistency	Fixed
F-2023-0608 - Double-Spending	Fixed
F-2023-0609 - Contradiction	Fixed
F-2023-0610 - Floating Pragma	Fixed
F-2023-0611 - Shadowing State Variable	Fixed
F-2023-0612 - Unindexed Events	Fixed

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 Bloqhouse Technologies B.V.
Audited By	Hacken
Website	http://www.bloqhouse.com/
Changelog	13/03/2023 - Initial Review 24/03/2023 - Second Review

Table of Contents

System Overview

The project is a real world asset tokenization protocol, which aims to enable asset providers to fractionalize assets such as, but not limited to, real estate, land, property, on the blockchain as NFTs.

RWATP (Real World Asset Tokenization Protocol) is an ERC721 based tokenization protocol. One instance of RWATP can have multiple assets, this enables one asset provider to handle several assets in one contract. An asset is a set of NFTs, one NFT represents a share of the asset. E.g if an asset consists of 100 NFTs then one NFT represents 1% of the asset.

The contract admin (asset provider or someone else) can decide how to mint and distribute the NFTs, either through minting the NFTs to the asset provider, to a user, or delegating the minting to a separate contract that can then add additional logic to the distribution process, for example crypto payment.

The supply of an asset is not fixed, but controlled by the contract admin. The contract enables issuance of new NFTs for an asset, thus diluting the supply, at their discretion. Likewise the admin can enable burning of NFTs, concentrating the supply. However, these features can be locked in perpetuity to ensure a constant number of NFTs for an asset.

The admin of the contract has rights to the funds such as burning, minting, pausing transfers, setting whitelist functionality(only allowing whitelisted addresses to transfer their tokens), and transferring them from users. This functionality can be locked by the admin to never be enabled again.

The files in the scope:

- **RWAT.sol**: The ERC721 contract that represents the assets. Can be controlled by the admin.
- **ISNR.sol**: The interface for obtaining the Token URI.

Privileged roles

- *DEFAULT_ADMIN_ROLE*: The DEFAULT_ADMIN_ROLE is inherited from the OpenZeppelin AccessControl.sol contract. It is the role that can grant or revoke all other roles. Presumably this should be given to a secure multisig and not used for anything other than granting/revoking the other roles.
- *Admin*: The ADMIN role controls all issuance, burning and settings of the contract.
- *Handler*: HANDLER can be given to a separate contract adding additional logic to the minting process. This role can only control the minting of new NFTs.
- *Whitelister*: There is one role for adding users to the whitelist, this is separate from the admin role to add an extra level of security.
- *User*: A user has no special privileges unless the whitelist is enabled, then the user must be on the whitelist in order to do anything other than holding the NFT.

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 comprehensive.
- Technical description is detailed.
- NatSpec is consistent.

Code quality

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

- Solidity best practices are followed.
- Style guides are followed.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is present.
- Interactions with several users are tested thoroughly.

Security score

Upon auditing, the code was found to contain **0** critical, **4** high, **0** medium, and **3** low severity issues. Out of these, **7** issues have been addressed and resolved, 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 admin has comprehensive rights to user funds which makes the project centralized. These rights include forcing funds out of wallets without a user's permission, pausing transfers, changing name, symbol, and asset caps.
- The contracts are upgradable and the current audit covers only the implementation at the time of the review. Any further implementations are not covered by this audit.

Findings

Vulnerability Details

F-2023-0606 - EIP Standard Violation - High

Description: Signatures do not include chain-specific parameters like chain id as stated in the EIP-712 standard.

This may lead to signature replay attacks if the contract is deployed multiple times and the verifier address is the same.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: High

Recommendations

Remediation: Follow the EIP-712 standard when creating and verifying signatures.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0607 - Data Consistency - High

Description: The project uses the tokens for every asset to determine the percentage of the asset the token holder holds. The contract *RWAT.sol* allocates 1.000.000.000 ids for every asset. The `_tokenCap` is not checked according to this allocation.

If the `_tokenCap` is larger than the max allocation of 1.000.000.000, the tokenIds might override the next asset allocation.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: High

Recommendations

Remediation: Verify that the `_tokenCap` is smaller than 1.000.000.000 when setting or updating asset caps.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0608 - Double-Spending - High

Description: In the `claimUnits()` function, there is no internal nonce check for the signature verification. In the scenario that:

- the `reclaimUnitsDisabled` parameter is true,
- The admin gets all tokens back (specifically the ones that were claimed previously), and transfers them back into the contract;

the user can successfully call the `claimUnits()` function with the same signature and transfer the tokens to their address again. This would lead to fund loss since the admin cannot reclaim the tokens.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: High

Recommendations

Remediation: Add a nonce mechanism for this function specific for the claiming users into the signature and increment it after the call so that the same signature cannot be used.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0609 - Contradiction - High

Description:

The project intends to override the ERC721 `_name` and `_symbol` parameters; however since it is using different names which are `name_` and `symbol_` for *RWAT.sol*, the parameters are not actually overwritten. The functions to view them are overwritten, which handles the issue but the function `setNameAndSymbol()` should be called before it is actually active.

This function is not called in the `initializer()` of the contract, only the `ERC721Upgradeable` parameters are set. This will cause the contract to be initialized without any name or symbol.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: High

Recommendations

Remediation: Call the `setNameAndSymbol()` function in the `initializer()`.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0610 - Floating Pragma - Low

Description: The project uses floating pragma ^0.8.4 in contracts *RWAT.sol* and *ICNR.sol*.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]
- interfaces/ICNR.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0611 - Shadowing State Variable - Low

Description: In RWA_.sol_ contracts' `initialize()`, `setTransfersPaused()`, `setAssetTransfersPaused()`, and the `setNameAndSymbol()` function, variables `_name`, `_symbol`, and `_paused` are shadowed from the *ERC721Upgradeable* contract.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Rename related variables/arguments.

Resolution: The Finding was fixed in commit cbdc7c0.

F-2023-0612 - Unindexed Events - Low

Description: Having indexed parameters in the events makes it easier to search for these events using indexed parameters as filters.

Assets:

- contracts/RWAT.sol [https://bitbucket.org/alfredpersson/token-shares-solidity/src/master]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Use the “indexed” keyword to the event parameters.

Resolution: The Finding was fixed in commit cbdc7c0.

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://bitbucket.org/alfredpersson/token-shares-solidity/src/master
Commit	cbdc7c0d6162346b96cf62cb2ff93c15f416819e
Whitepaper	Not provided
Requirements	Provided
Technical Requirements	Provided

Contracts in Scope

contracts/RWAT.sol

interfaces/ICNR.sol