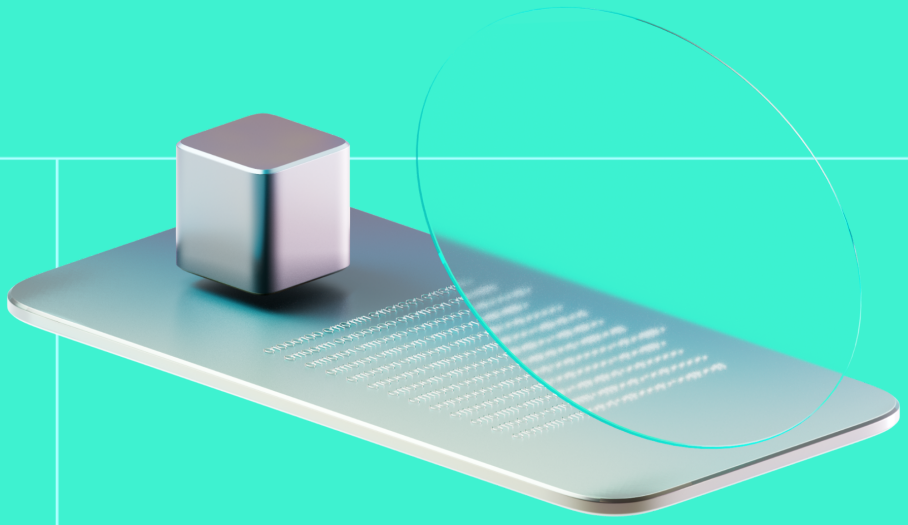
HACKEN

# Smart Contract Code Review And Security Analysis Report



**Customer:** Hex Trust

**Date:** 15 Dec, 2023

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

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

**Platform**: EVM

**Timeline**: 20.11.2023 - 15.12.2023

**Language**: Solidity

**Methodology**: Link

**Tags**: ERC-20

## Last review scope

| | |
|---|---|
| **Repository** | https://github.com/HTMI-Ltd/hex-stablecoin-usd |
| **Commit** | ccefe903cf8ea2e762216686f69d197bb6ab9cb5 |

View full scope

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

## Audit Summary

| 10/10 | 10/10 | 90.63% | 10/10 |
|---|---|---|---|
| Security score | Code quality score | Test coverage | Documentation quality score |

**Total: 9.6/10**

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 |

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



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Hex Trust |
| **Approved By** | Przemyslaw Swiatowiec \| SC Audits Expert at Hacken OÜ |
| **Audited By** | Carlo Parisi \| SC Lead Auditor at Hacken OÜ<br>Roman Tiutiun \| SC Auditor at Hacken OÜ |
| **Changelog** | 24.11.2023 – Preliminary Report<br>15.12.2023 – Second Review |

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

## Introduction

Hacken OÜ (Consultant) was contracted by Hex Trust (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

Hex Trust is a Hex Stablecoin (HUSD):

- HexStableCoin — ERC20 upgradeable token.
- AccessControlDefaultAdminRulesUpgradeable — Fork of an OpenZeppelin contract with a few modifications.
- PausableWithRolesUpgradeable — allows contract to be paused and upaused by the *PAUSER_ROLE*.
- ERC20WithRolesUpgradeable — ERC-20 functionalities along with supply control functions (minting and burning) also introduced a function to seize the Token of a blacklisted party when required.
- BlacklistableWithRolesUpgradeable — contract to allow and remove blacklisted users in the contract. Only *BLACKLISTER_ROLE* can add or remove blacklisted users.
- RoleConstant — hashed string values for each role.

**Privileged roles**

- PAUSER_ROLE: can pause and unpause the contract.

- MINTER_ROLE: can mint unlimited tokens.
- BLACKLISTER_ROLE: add to and remove users from the blacklist.

The role can also transfer tokens from blacklisted users addresses.

- DEFAULT_ADMIN_ROLE: can reassign roles.
- BURNER_ROLE: can burn tokens from any address.
- UPGRADE_ADMIN_ROLE: can upgrade the upgradeable contracts.

# 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 provided.
- Natspec provided.
- Technical requirements provided.

### Code quality

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

### Test coverage

Code coverage of the project is **90.63%** (branch coverage).

### Security score

No issues were found in the audited code. The security score is **10** out of **10**.

Informational observations are displayed in the “Findings” section.

### Summary

According to the assessment, the Customer's smart contract has the following score: **9.6**. The system users should acknowledge all the risks summed up in the risks section of the report.

## Risks

- Contracts can be upgraded after deployment. These changes must be approached with caution as they can potentially introduce critical vulnerabilities.
- The contract owner possesses significant control over the contracts, which includes the ability to pause, unpause, blacklist users, transfer funds from blacklisted accounts, burn funds from any wallet, and mint new tokens.
- The contract owner will be responsible for the appropriate minting of the stablecoin, the functionality of *mint()* allows the *MINTER_ROLE* an unlimited capacity to mint new tokens.

# 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

### I01. Floating pragma

All contracts use the *floating pragma ^0.8.20*.

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.

**Path:** ./contracts/*

**Recommendation**: Lock the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

**Found in**: f82212f

**Status:** Fixed (Revised commit: 5e4ec54)

**Remediation:** Solidity Pragma version was locked in contracts.

### I02. Upgradeable contracts not initialized

During the initialization of the *HexStableCoin* contract, only the inherited contracts *UUPSUpgradeable*, *ERC20WithRolesUpgradeable*, *AccessControlDefaultAdminRulesUpgradeable*, *PausableWithRolesUpgradeable* and *BlacklistableWithRolesUpgradeable* undergo initialization.

However, it is important to note that *PausableUpgradeable*, *AccessControlUpgradeable* are not being initialized.

While this lack of initialization does not pose a security vulnerability, adhering to best practices suggests initializing all upgradeable contracts to ensure a consistent and robust codebase.

**Path:** ./contracts/HexStableCoin.sol

**Recommendation**: It is recommended to align with convention and initialize all upgradeable contracts.

**Found in**: f82212f

**Status:** Fixed (Revised commit: fda4a0)

**Remediation:** Inherited contract *UUPSUpgradeable*, *ERC20WithRolesUpgradeable*, *AccessControlDefaultAdminRulesUpgradeable*,

*PausableWithRolesUpgradeable* and *BlacklistableWithRolesUpgradeable* are initialized.

# 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/HTMI-Ltd/hex-stablecoin-usd |
| Commit | f82212fa1bef11f248f08488614f048f4d807b8a |
| Requirements | https://github.com/HTMI-Ltd/hex-stablecoin-usd |
| Technical Requirements | https://github.com/HTMI-Ltd/hex-stablecoin-usd |

### Contracts in Scope

File: contracts/AccessControlDefaultAdminRulesUpgradeable.sol
File: contracts/BlacklistableWithRolesUpgradeable.sol
File: contracts/ERC20WithRolesUpgradeable.sol
File: contracts/HexStableCoin.sol
File: contracts/PausableWithRolesUpgradeable.sol
File: contracts/utils/Context.sol
File: contracts/utils/RoleConstant.sol

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

## Scope details 2

| | |
|---|---|
| Repository | https://github.com/HTMI-Ltd/hex-stablecoin-usd |
| Commit | ccefe903cf8ea2e762216686f69d197bb6ab9cb5 |
| Requirements | https://github.com/HTMI-Ltd/hex-stablecoin-usd |
| Technical Requirements | https://github.com/HTMI-Ltd/hex-stablecoin-usd |

## Contracts in Scope 2

File: contracts/AccessControlDefaultAdminRulesUpgradeable.sol
File: contracts/BlacklistableWithRolesUpgradeable.sol
File: contracts/ERC20WithRolesUpgradeable.sol
File: contracts/HexStableCoin.sol
File: contracts/PausableWithRolesUpgradeable.sol
File: contracts/utils/Context.sol
File: contracts/utils/RoleConstant.sol