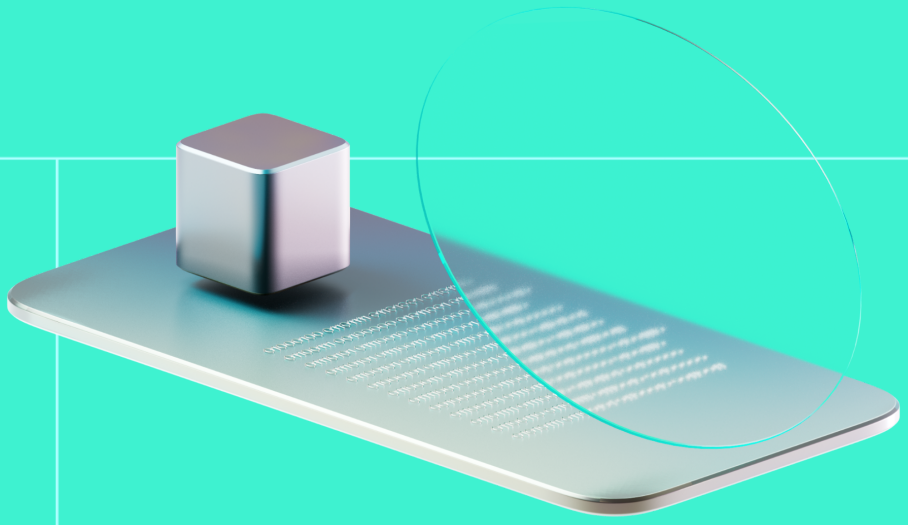
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Minteo

Date: 10 Nov, 2023

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

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

Minteo creates blockchain based infrastructure to stimulate trade and enable new business models in Latin America. Minteo has the mission of providing a stable and reliable digital Colombian Peso: COPM, a fully collateralized stablecoin backed with highly liquid assets denominated in Colombian Pesos, held in reserve.

Platform: EVM

Language: Solidity

Tags: ERC-20, UUPSProxy, Stablecoin

Timeline: 07.11.2023 - 14.11.2023

Methodology: Link

Last review scope

Repository	https://github.com/minteo-wagmi/rwa-contracts
Commit	985b0f30c05f1f801030e4d06282a3c7eb111917

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

Audit Summary

10/10	9/10	100%	9/10
Security score	Code quality score	Test coverage	Documentation quality score

Total: 9.5/10

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

1	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	1	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 Minteo
Approved By	Paul Fomichov | SC Audits Expert at Hacken OÜ
Audited By	Grzegorz Trawiński | SC Audits Expert at Hacken OÜ
Website	https://minteo.com/
Changelog	14.11.2023 – Preliminary Report

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

Introduction

Hacken OÜ (Consultant) was contracted by Minteo (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

TokenV1 is a stablecoin solution with the following contracts:

- TokenV1 — is an ERC-20 token with centralized authorization. Version 1 removes the possibility of flash loans. The contract is upgradable by means of UUPS Proxy. This contract inherits following OpenZepplin's contracts:
    - Initializable,
    - ERC20Upgradeable,
    - ERC20BurnableUpgradeable,
    - PausableUpgradeable,
    - AccessControlEnumerableUpgradeable,
    - ERC20PermitUpgradeable,
    - StorageGaps,
    - UUPSUpgradeable,
    - Freezable,
    - MulticallUpgradeable

It has the following attributes:

- Name: COP Minteo
- Symbol: COPM
- Decimals: 18
- Total supply: not defined.

- Freezable — a contract that allows to temporarily froze an account's funds.
- StorageGaps — a contract that reserves storage space after removing ERC20FlashMintUpgradeable from inheritance tree.

Privileged roles

- The solution supports following roles:
  - Administrator
  - Pauser
  - Minter
  - Freezer
  - Upgrader
  - Rescuer
- The administrator role of the TokenV1 contract can manage roles within the solution.
- The pauser role of the TokenV1 contract can pause and unpause the solution

- The minter role of the TokenV1 contract can mint tokens
- The freezer role of the TokenV1 contract can freeze and unfreeze an account's funds.
- The upgraded role of the TokenV1 contract can upgrade the implementation of UUPS Proxy.
- The rescuer role of the TokenV1 contract can withdraw ERC20 tokens sent to the contract.

Minteo utilizes multi-signature wallets to handle every privileged account.

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 **9** out of **10**.

- Functional requirements were provided.
- Technical requirements were not provided.
- However, mostly, code is self-documented.

Code quality

The total Code Quality score is **9** out of **10**.

- The code inherits mostly from OpenZeppling extensions
- Code follows common best practices.
- The development environment is configured.
- Mostly, code is self-documented.

- No NatSpec documentation provided.

Test coverage

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

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage are included.
- Interactions by several users are included.
- The Upgradability functionality is tested as well.
- Tests use Polygon fork.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.5**. The system users should acknowledge all the risks summed up in the risks section of the report.

Risks

- The TokenV1 solution is heavily centralized. Apart from the standard ERC20 functionality, every public function requires authorisation of privileged accounts. Minteo states that every account is a multi-signature wallet.
- The solution owner can freeze any account’s funds.
- The solution is upgradable by means of UUPS Proxy.

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

L01. Lack of two-step ownership transfer

Impact	Low
Likelihood	Low

The TokenV1 contract inherits from OpenZepplin's *AccessControlEnumerableUpgradeable*, which inherits from the *AccessControlUpgradeable*. This extension allows the use of role-based authorization with the possibility of enumerating the members of each role. However, OpenZepplin offers the *AccessControlDefaultAdminRulesUpgradeable* extension as well, which has additional security-related benefits implemented:

- Only one account can hold the `*DEFAULT_ADMIN_ROLE*` since deployment until it is potentially renounced.

- Enforces a 2-step process to transfer the `*DEFAULT_ADMIN_ROLE*` to another account.
- Enforces a configurable delay between the two steps, with the ability to cancel before the transfer is accepted. The delay is also configurable.

Thus, the current implementation of TokenV1 is missing a two-step ownership-transfer process among the others. As a result, in case of mistaken ownership transfer to the invalid account, all administrative functionalities may become unavailable for contract owners.

```
contract TokenV1 is
 Initializable,
 ERC20Upgradeable,
 ERC20BurnableUpgradeable,
 PausableUpgradeable,
 AccessControlEnumerableUpgradeable,
 ERC20PermitUpgradeable,
 StorageGaps,
 UUPSUpgradeable,
 Freezable,
 MulticallUpgradeable
{
```

Path: ./src/TokenV1.sol

Recommendation: It is recommended to consider changing the inheritance structure of the TokenV1 smart contract, so it inherits from *AccessControlDefaultAdminRulesUpgradeable* and apply the enumerable functionality on top of that.

Found in: 985b0f

Status: New

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

Informational

No informational observations 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/minteo-wagmi/rwa-contracts
Commit	985b0f30c05f1f801030e4d06282a3c7eb111917
Whitepaper	[Undisclosed]
Requirements	[Undisclosed]
Technical Requirements	N/A

Contracts in Scope

./src/TokenV1.sol ./src/UUPSProxy.sol ./src/Freezable.sol ./src/StorageGaps.sol