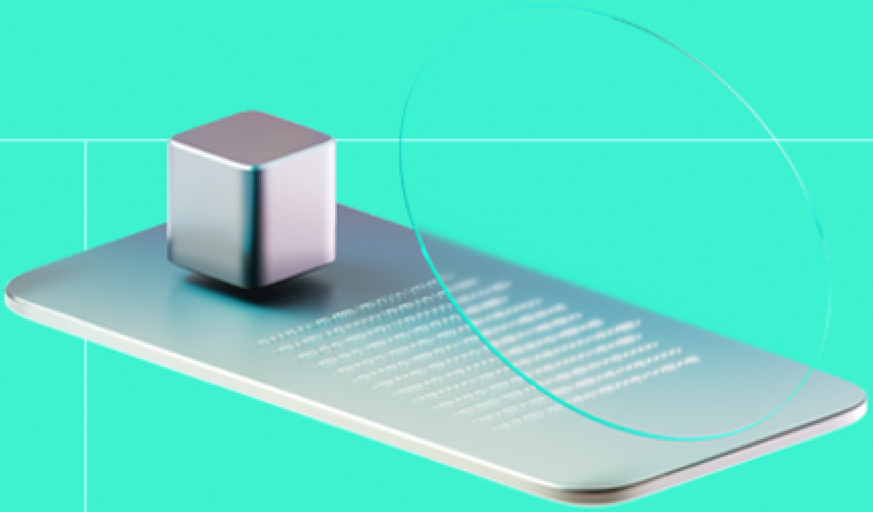
Smart Contract Code Review And Security Analysis Report



Customer: Cryptopia

Date: 22/02/2024

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

Cryptos Token, the token within a blockchain game in development, serves the purpose of facilitating seamless interoperability across multiple blockchains.

Platform: EVM

Language: Solidity

Tags: ERC20, Gaming

Timeline: 15/02/2024 - 16/02/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/cryptopia-com/cryptopia-token-contracts
Commit	43b3561

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

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

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 Cryptopia
Audited By	Philipp Eder
Approved By	Yves Toiser
Website	https://hacken.io
Changelog	20/02/2024 - Preliminary Report & 22/02/2024 - Final Report

Table of Contents

System Overview

Cryptopia is an upcoming blockchain game.

CryptosToken — simple ERC-20 token that mints all initial supply to a deployer. It is intended to have implementations on both Ethereum Mainnet and Polygon Mainnet and comes with functionality to allow seamless bridging.

It has the following attributes:

- Name: Cryptos
- Symbol: TOS
- Decimals: 18
- Total supply: 10 billion tokens.

CryptosTokenPolygon - the implementation of CryptosToken on the Polygon blockchain.

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**.

Code quality

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

Test coverage

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

Security score

Upon auditing, the code was found to contain **0** 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 security of the bridging functionality is dependent on the security of the polygon bridge which was not subject of examination within the scope of this audit.
- The owners need to be trusted to utilize the official polygon bridge as the depositor in CryptosTokenPolygon.sol
- The owners need to be trusted to facilitate the mapping of the rootChain contract and childChain contract correctly in order for the bridging process to work as intended.
- The version of Solidity used in this project might not work on all chains, due to the opcode push0, however it is supported by the chains it is intended to be used on.

Findings

Vulnerability Details

Observation Details

F-2024-0863 - Floating pragma - Info

Description: The project uses floating pragmas `^0.8.20 < 0.9.0`

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:

- cryptos/CryptosToken.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20Retriever.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/AccessErrors.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: It is recommended to lock the pragma.
e.g. pragma solidity 0.8.20;

Remediation: The client has fixed this observation.

F-2024-0865 - CryptosTokens will be locked inside the contract if sent by accident - Info

Description: The overridden retrieveTokens() function will revert if the admin tries to recover CryptosTokens that were accidentally sent to the contract address.

Assets:

- cryptos/CryptosToken.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: Remove the conditional statement within the overridden function to make sure that all ERC20 tokens can be recovered.

Remediation: The client has fixed this observation.

F-2024-0922 - Missing event emission - Info

Description:

The contract CryptosTokenPolygon.sol lacks events to track important operations like deposits or withdraws.
The contract CryptopiaERC20.sol lacks an event emission to track important operations like retrieving 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:

- cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

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

- For `deposit()`, emit an event detailing the beneficiary address and amount.
- For `withdraw()`, emit an event capturing both the user's address and amount.
- For `retrieveTokens()`, emit an event capturing the token's address, and amount.

Remediation: The client has fixed this observation.

F-2024-0923 - Variable shadowing in CryptopiaERC20.sol - Info

Description: The variables name and symbol in CryptopiaERC20.sol shadow the variables in the inherited ERC20.sol contract.

Assets:

- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: Rename the variables by adding a prefixed underscore.

example: _name, _symbol

Remediation: The client has fixed this observation.

F-2024-0928 - Variables only set in constructor() should be marked as immutable - Info

Description: The variable depositor in the CryptosTokenPolygon.sol should be set as immutable.

Assets:

- cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: **Remediation**: The client has fixed this observation.

F-2024-0929 - Missing zero address checks - Info

Description:	There is no zero address check for assigning the depositor address in the constructor and the user address within the deposit() function.
Assets:	• cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
Status:	Fixed

Recommendations

Recommendation:	Implement a zero address check for the depositor address in constructor(). Implement a zero address check for the user address in deposit(). **Remediation**: The client has fixed this observation.

F-2024-0930 - Functions not called internally can be marked as external - Info

Description: Functions that are meant to be exclusively invoked from external sources should be designated as `external` rather than `public`. This is beneficial for gas efficiency and the overall security of the contract.

Assets:

- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20Retriever.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: The functions retrieveTokens() in the CryptopiaERC20.sol and CryptopiaERC20Retriever.sol can be reduced to “external” visibility.

Remediation: The client has fixed this observation.

F-2024-0931 - Redundant contract structure - CryptopiaERC20Retriever.sol - Info

Description: The contract CryptopiaERC20Retriever.sol consists solely of the retrieveTokens() function which is overridden in the contracts using it.

Assets:

- cryptos/CryptopiaERC20.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/CryptopiaERC20Retriever.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Fixed

Recommendations

Recommendation: Remove the contract CryptopiaERC20Retriever.sol and instead include the function retrieveTokens() in the CryptopiaERC20.sol contract to reduce deployment cost and simplify the contract structure.

Remediation: The client has fixed this observation.

F-2024-0932 - Redundant contract structure - AccessErrors.sol - Info

Description: The contract AccessErrors.sol consists solely of the Unauthorized() error which is only used in CryptosTokenPolygon.sol.

Assets:

- cryptos/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]
- cryptos/AccessErrors.sol [https://github.com/cryptopia-com/cryptopia-token-contracts]

Status: Accepted

Recommendations

Recommendation: Remove the file AccessErrors.sol and instead include the error Unauthorized() in the CryptosTokenPolygon.sol contract to reduce deployment cost and simplify the contract structure.

Remediation: The client has accepted 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://github.com/cryptopia-com/cryptopia-token-contracts
Commit	43b35619a25616c6873d384891a499c13f8af03f
Whitepaper	
Requirements	
Technical Requirements	

Contracts in Scope

./contracts/source/CryptopiaERC20.sol

./contracts/source/errors/AccessErrors.sol

./contracts/source/errors/ArgumentErrors.sol

./contracts/source/ethereum/CryptosToken.sol

./contracts/source/polygon/CryptosTokenPolygon.sol