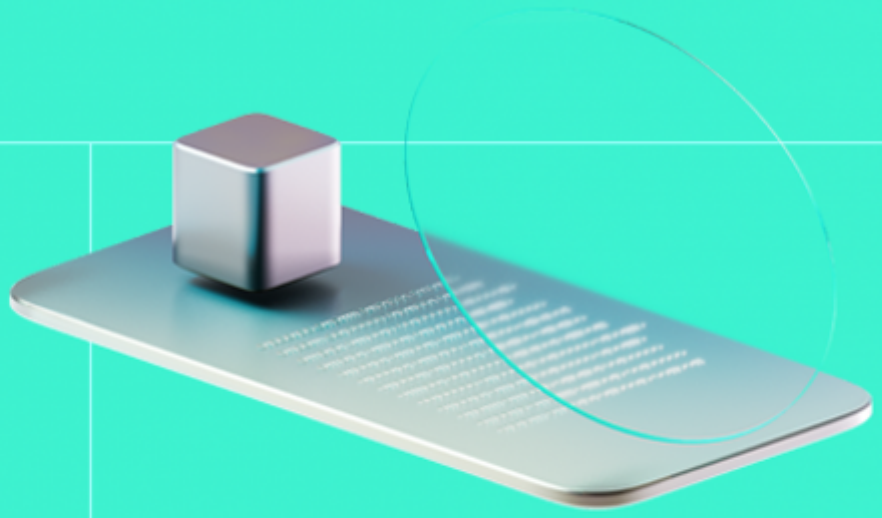
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Cryptopia

Date: 11/04/2024

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

Cryptopia, is an innovative blockchain-based game pioneering the future of decentralized, play-to-earn gaming. Cryptopia designed the smart contract OFT-based bridging solution to power the token functionalities.

Platform: Ethereum, Polygon

Language: Solidity

Tags: OFT, Fungible Token, Gamify, Bridge

Timeline: 15/03/2024 - 11/04/2024

Methodology: https://hackenio.cc/sc_methodology

Review

Scope

Repository	https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero
Commit	e476da911af21a941d2f63b411bd2d943b40338c

Audit Summary

10/10	10/10	92,86%	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	0
High	0
Medium	0
Low	1

Vulnerability	Status
F-2024-1538 - Lack of two-step ownership transfer	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 Cryptopia
Audited By	Grzegorz Trawinski
Approved By	Ataberk Yavuzer
Website	https://cryptopia.com/
Changelog	19/03/2024 - Preliminary Report
	11/04/2024 - Final Report

Table of Contents

System Overview

The solution is an OFT-based bridge between Ethereum and Polygon blockchains. It uses OFT V2 version.

The CryptosToken is an ERC20 token representing the game currency used in Cryptopia. It supports ERC20 tokens retrieval accidentally sent to the contract. It has following characteristics:

- Name: Cryptos
- Symbol: TOS
- Decimals: 18
- Supply: 1e28

The CryptosTokenOFTAdapter is an OFTAdapter. It allows the existing token to expand to any supported chain as a native token with a unified global supply, inheriting all the features of the OFT Standard. This works as an intermediary contract that handles sending and receiving tokens that have already been deployed.

The CryptosTokenPolygon is an OFT-based (ERC20) token to be deployed on Polygon blockchain. It supports ERC20 tokens retrieval accidentally sent to the contract. This contract allows to deposit tokens for a user by Polygon Bridge Depositor and withdraw tokens by a user.

Privileged roles

- The owner of the CryptosToken contract can retrieve ERC20 tokens sent to the contract.
- The owner of the CryptosTokenPolygon contract can retrieve ERC20 tokens sent to the contract. Also, it has access to the OFT privileged actions.
- The Polygon Bridge Depositor of the CryptosTokenPolygon contract can mint tokens for a user.
- The owner of the CryptosTokenOFTAdapter contract has access to the OFT privileged actions.

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 **92.86%** (branch coverage).

Security score

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

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 **10.0**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The solutions uses LayerZero, an interoperability protocol, for ERC20 token bridging purposes. Thus, tokens are bridged and minted by the third party privileged entities.

Findings

Vulnerability Details

F-2024-1538 - Lack of two-step ownership transfer - Low

Description:

The solution implements single step ownership transfer. Thus, accidental transfer of ownership to unverified and incorrect address may result in loss of ownership. In such a case, access to every function protected by the ownership check will be permanently lost.

The `CryptosToken` contracts inherits OpenZepplin's `Ownable` directly. The `CryptosTokenOFTAdapter` and `CryptosTokenPolygon` inherit `Ownable` indirectly from the OFT components

```
. contract CryptosToken is ERC20, Ownable, CryptopiaTokenRetriever {
```

```
contract CryptosTokenOFTAdapter is OFTAdapter {
...
constructor(address _token,address _layerZeroEndpoint, address _owne
r )
OFTAdapter(_token, _layerZeroEndpoint, _owner) Ownable(_owner) {}
}
```

```
contract CryptosTokenPolygon is OFT, CryptopiaTokenRetriever {
...
constructor(address _polygonBridgeDepositor, address _layerZeroEndpo
int, address _owner )
OFT("Cryptos", "TOS", _layerZeroEndpoint, _owner) Ownable(_owner)
```

Assets:

- /contracts/ethereum/CryptosToken.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]
- /contracts/ethereum/CryptosTokenOFTAdapter.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]
- /contracts/polygon/CryptosTokenPolygon.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]

Status: Fixed

Classification

Severity: Low

Impact: Likelihood [1-5]: 1
Impact [1-5]: 4

Exploitability [0-2]: 0
Complexity [0-2]: 0
Final Score: 2.0 (Low)

Recommendations

Remediation: It is recommended to implement a two-step ownership transfer pattern within the solution, such as OpenZepplin's `Ownable2Step`.

Remediation (commit Id: 901e8acc2bf0c77076b5f0dc47ba797d9b5ed2ff): The two-step ownership transfer pattern is now implemented.

Observation Details

F-2024-1559 - Default sharedDecimals() with 6 value is in use - Info

Description:

The solution integrates with the LayerZero interoperability protocol that acts as bridge between chains. It was identified that in each case the solution uses the default `sharedDecimals()` implementation with value set to 6, where the `CryptosToken` and `CryptosTokenPolygon` are ERC20 implementations that uses 18 decimals.
Such implementation results in possibility of transferring tokens amounts only with top 6 decimal places, where everything smaller cannot be transferred.

```
function sharedDecimals() public pure virtual returns (uint8) {
return 6;
}
```

Assets:

- /contracts/ethereum/CryptosToken.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]
- /contracts/ethereum/CryptosTokenOFTAdapter.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]
- /contracts/infrastructure/CryptopiaTokenRetriever.sol [https://github.com/cryptopia-com/cryptopia-token-contracts/tree/layerzero/]

Status: Fixed

Recommendations

Remediation:

It is recommended to review the design of the solution and consider changing the value set within the `sharedDecimals()` function to satisfy the business requirement rules.

Remediation: The Client's team confirmed that they do consider usage of non-EVM based blockchains in future releases. Thus, the current configuration is consistent with business requirements.

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/tree/layerzero
Commit	e476da911af21a941d2f63b411bd2d943b40338c
Whitepaper	N/A
Requirements	https://github.com/cryptopia-com/cryptopia-token-contracts/blob/layerzero/docs/Cryptos%20-%20Audit%20Techspec.pdf
Technical Requirements	https://github.com/cryptopia-com/cryptopia-token-contracts/blob/layerzero/docs/Cryptos%20-%20Audit%20Techspec.pdf

Contracts in Scope

./source/polygon/CryptosTokenPolygon.sol

./source/errors/AccessErrors.sol

./source/errors/ArgumentErrors.sol

./source/infrastructure/CryptopiaTokenRetriever.sol

./source/errors/AccessErrors.sol

./source/errors/ArgumentErrors.sol

./source/infrastructure/CryptopiaTokenRetriever.sol

./source/ethereum/CryptosTokenOFTAdapter.sol

./source/ethereum/CryptosToken.sol