EVA Coin

SECURITY ASSESSMENT REPORT

September 23, 2025

Prepared for

Contents

1 About CODESPECT

CODESPECT is a specialized smart contract security firm dedicated to ensure the safety, reliability, and success of blockchain projects. Our services include comprehensive smart contract audits, secure design and architecture consultancy, and smart contract development across leading blockchain platforms such as Ethereum (Solidity), Starknet (Cairo), and Solana (Rust).

At CODESPECT, we are committed to build secure, resilient blockchain infrastructures. We provide strategic guidance and technical expertise, working closely with our partners from concept development through deployment. Our team consists of blockchain security experts and seasoned engineers who apply the latest auditing and security methodologies to help prevent exploits and vulnerabilities in your smart contracts.

Smart Contract Auditing: Security is at the core of everything we do at CODESPECT. Our auditors conduct thorough security assessments of smart contracts written in Solidity, Cairo, and Rust, ensuring that they function as intended without vulnerabilities. We specialize in providing tailored security solutions for projects on EVM-compatible chains and Starknet. Our audit process is highly collaborative, keeping clients involved every step of the way to ensure transparency and security. Our team is also dedicated to cutting-edge research, ensuring that we stay ahead of emerging threats.

Secure Design & Architecture Consultancy: At CODESPECT, we believe that secure development begins at the design phase. Our consultancy services offer deep insights into secure smart contract architecture and blockchain system design, helping you build robust, secure, and scalable decentralized applications. Whether you're working with Ethereum, Starknet, or other blockchain platforms, our team helps you navigate the complexity of blockchain development with confidence.

Tailored Cybersecurity Solutions: CODESPECT offers specialized cybersecurity solutions designed to minimize risks associated with traditional attack vectors, such as phishing, social engineering, and Web2 vulnerabilities. Our solutions are crafted to address the unique security needs of blockchain-based applications, reducing exposure to attacks and ensuring that all aspects of the system are fortified.

With a focus on the intersection of security and innovation, CODESPECT strives to be a trusted partner for blockchain projects at every stage of development and for each aspect of security.

2 Disclaimer

Limitations of this Audit: This report is based solely on the materials and documentation provided to CODESPECT for the specific purpose of conducting the security review outlined in the Summary of Audit and Files. The findings presented in this report may not be comprehensive and may not identify all possible vulnerabilities. CODESPECT provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, is entirely at your own risk.

Inherent Risks of Blockchain Technology: Blockchain technology is still evolving and is inherently subject to unknown risks and vulnerabilities. This review focuses exclusively on the smart contract code provided and does not cover the compiler layer, underlying programming language elements beyond the reviewed code, or any other potential security risks that may exist outside of the code itself.

Purpose and Reliance of this Report: This report should not be viewed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. Third parties should not rely on this report for any purpose, including making decisions related to investments or purchases.

Liability Disclaimer: To the maximum extent permitted by law, CODESPECT disclaims all liability for the contents of this report and any related services or products that arise from your use of it. This includes but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services: CODESPECT does not warrant, endorse, or assume responsibility for any third-party products or services mentioned in this report, including any open-source or third-party software, code, libraries, materials, or information that may be linked to, referenced by, or accessible through this report. CODESPECT is not responsible for monitoring any transactions between you and third-party providers. We strongly recommend conducting thorough due diligence and exercising caution when engaging with third-party products or services, just as you would for any other product or service transaction.

Further Recommendations: We advise clients to schedule a re-audit after any significant changes to the codebase to ensure ongoing security and reduce the risk of newly introduced vulnerabilities. Additionally, we recommend implementing a bug bounty program to incentivize external developers and security researchers to identify and disclose potential vulnerabilities safely and responsibly.

Disclaimer of Advice: FOR AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, AND ANY ASSOCIATED SERVICES OR MATERIALS SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Risk Classification

Severity Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Table 1: Risk Classification Matrix based on Likelihood and Impact

3.1 Impact

- **High** - Results in a substantial loss of assets (more than 10%) within the protocol or causes significant disruption to the majority of users.
- **Medium** - Losses affect less than 10% globally or impact only a portion of users, but are still considered unacceptable.
- **Low** - Losses may be inconvenient but are manageable, typically involving issues like griefing attacks that can be easily resolved or minor inefficiencies such as gas costs.

3.2 Likelihood

- **High** - Very likely to occur, either easy to exploit or difficult but highly incentivized.
- **Medium** - Likely only under certain conditions or moderately incentivized.
- **Low** - Unlikely unless specific conditions are met, or there is little-to-no incentive for exploitation.

3.3 Action Required for Severity Levels

- **Critical** - Must be addressed immediately if already deployed.
- **High** - Must be resolved before deployment (or urgently if already deployed).
- **Medium** - It is recommended to fix.
- **Low** - Can be fixed if desired but is not crucial.

In addition to High, Medium, and Low severity levels, CODESPECT utilizes two other categories for findings: **Informational** and **Best Practices**.

a) **Informational** findings do not pose a direct security risk but provide useful information the audit team wants to communicate formally.

b) **Best Practices** findings indicate that certain portions of the code deviate from established smart contract development standards.

4 Executive Summary

This document presents the results of a security assessment conducted by CODESPECT for Portal EVA. Portal EVA is a global digital marketplace that connects customers and suppliers worldwide.

The scope of this audit included the review of an ERC-20 token contract, its implementation, accompanying documentation, and the token-related information provided in the whitepaper.

The audit was performed using:

a) Manual analysis of the codebase.

CODESPECT found one point of attention, one classified as `Informational`. All of the issues are summarised in Table 2.

Organisation of the document is as follows:

- **Section 5** summarizes the audit.
- **Section 6** describes the token properties.
- **Section 7** presents the issues.
- **Section 8** discusses the documentation provided by the client for this audit.

Issues found:

Severity	Unresolved	Fixed	Acknowledged
Informational	0	1	0
Total	**0**	**1**	**0**

Table 2: Summary of Unresolved, Fixed, and Acknowledged Issues

5 Audit Summary

Audit Type	Security Review
Project Name	EVA Coin
Type of Project	ERC-20
Duration of Engagement	1 Day
Duration of Fix Review Phase	1 Day
Draft Report	September 19, 2025
Final Report	September 23, 2025
Repository	Not Applicable
Polygon Testnet address (Audit)	0x1D65Af937cdF1d963Db5b6407436EE8490ba0463
Polygon Testnet address (Final)	0x78222A21f4CCE8a4737fF558745aAf48A81153D6
Documentation Assessment	Not Applicable
Test Suite Assessment	Not Applicable
Auditors	talfao

Table 3: Summary of the Audit

5.1 Scope - Audited Files

	Contract	**LoC**
1	EVAcoinToken.sol	30
	Total	**30**

5.2 Findings Overview

	Finding	**Severity**	**Update**
1	The `permit(...)` call in `transferFromWithPermit(...)` can be frontrun	Info	Fixed

6 EVA Token Properties overview

This table overview contains general information about the token and its properties:

Property	Value	Description	Additional Information
Name	EVA coin	Human-readable name of the token	On testnet deployed with `EVA coin test`
Symbol	EVAc	Ticker symbol of the token	On testnet deployed with `EVAc.t`
Decimals	18	Number of decimal places used	Standard Ethereum token precision
Total Supply	200,000,000 EVA	Total number of tokens created at deployment	Initial supply is defined during deployment, the defined target in documentation is 200 milions
Receiver	Defined by the deployer	Address that receives the full token supply upon deployment	Recommended to be multi-sig wallet
Owner	-	The token contract does not inherit from Ownable	Enhances decentralization
Mintable	No	Indicates whether new tokens can be minted post-deployment	Prevents supply manipulation
Burnable	Yes	Indicates whether tokens can be burned or destroyed	The holder or allowed spender can burn the holder's balance
Upgradable	No	Indicates whether the token contract is upgradeable	Immutable contract code
Permit	Yes	Indicates if contract inherits permitable extension	The contract implements ERC2612 which allow gasless approvals

Table 4: EVA Token Properties

Furthermore the contract implement a custom function `transferFromWithPermit(...)` which updates allowance based on the supplied signed message and initiates transfer from the user to the supplied receiver:

```
function transferFromWithPermit(
    address from,
    address to,
    uint256 amount,
    uint256 deadline,
    uint8 v,
    bytes32 r,
    bytes32 s
) public virtual;
```

7 Issues

7.1 [Info] The `permit(...)` call in `transferFromWithPermit(...)` can be frontrun

File(s): `EVAcoinToken.sol`

Description: The EVA token implements a custom function `transferFromWithPermit(...)`:

```
function transferFromWithPermit(
    address from,
    address to,
    uint256 amount,
    uint256 deadline,
    uint8 v,
    bytes32 r,
    bytes32 s
) public virtual {
    permit(from, msg.sender, amount, deadline, v, r, s);
    transferFrom(from, to, amount);
}
```

This function allows setting an allowance via `permit(...)` and transferring tokens in a single transaction.

The known issue arises because `permit(...)` is an **external function**. An attacker can frontrun the transaction by executing `permit(...)` first, causing the subsequent `transferFromWithPermit(...)` call to revert.

Impact:

- This is primarily a **griefing vector**.
- A user could still execute `transferFrom(...)` after `permit(...)` has been called, therefore the issue was marked as informative issue.

Recommendation(s):

- Consider wrapping the `permit(...)` call in a `try-catch` block to prevent transaction reverts from frontrunning.
- Alternatively, document this behaviour clearly for end users.

Status: Fixed

Update from EVA team:

8 Evaluation of Provided Documentation

The EVA team provided the project documentation in two distinct forms:

- The official project whitepaper, which outlines the token allocations, distribution mechanisms, and overall tokenomics.
- An internal documentation document, which provides detailed specifications for all token properties, including functional and operational aspects.

The CODESPECT team conducted a thorough review of both documents to assess whether the smart contract code implementation accurately reflects the specifications outlined in the documentation. This review included verifying that all described functionality is properly implemented, identifying any missing or inconsistent features, and ensuring that the code aligns with the intended token behaviour and design.