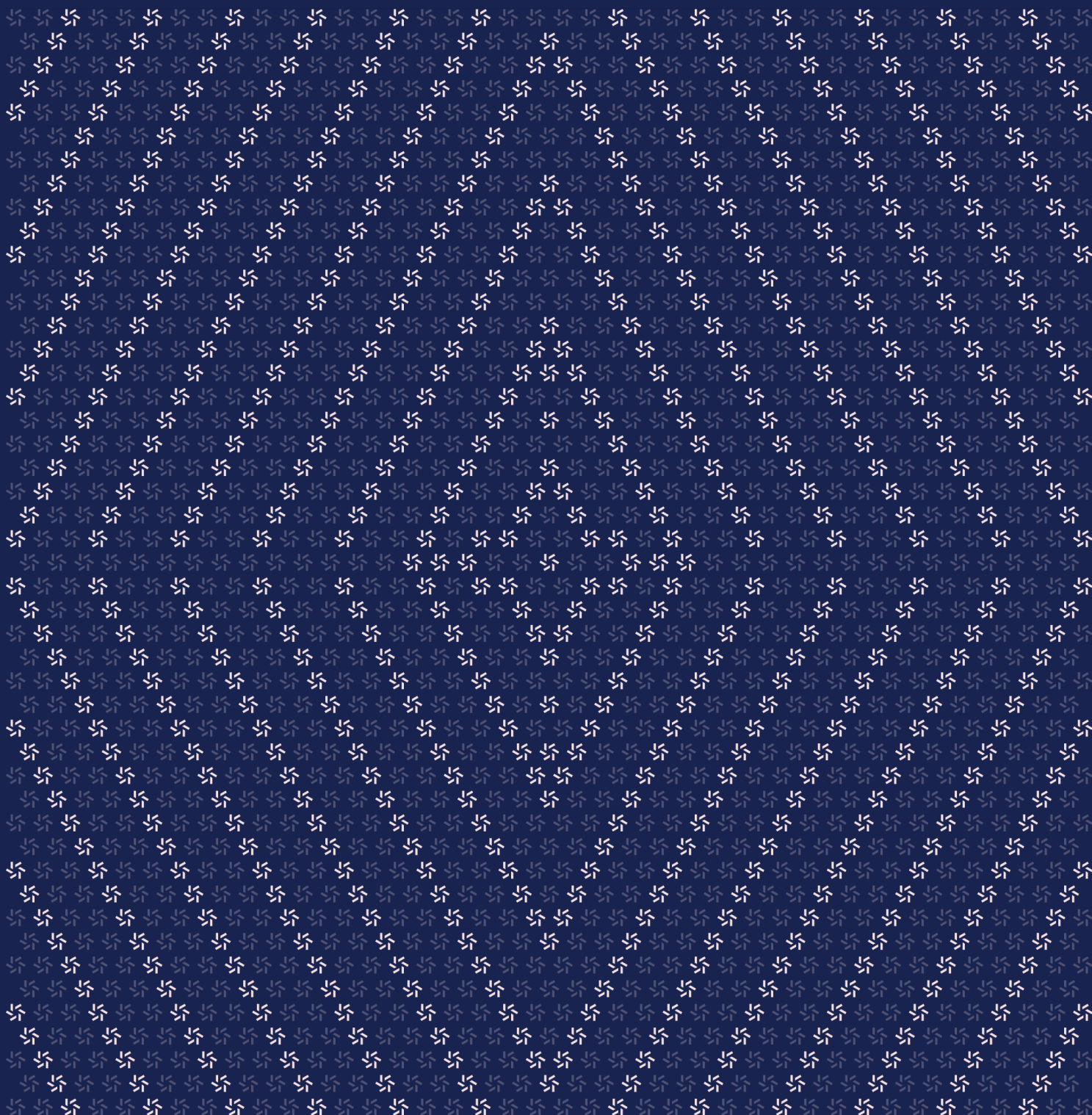


October 21, 2025

Agora Cross-Chain (LayerZero)

Smart Contract Security Assessment



Contents

About Zellic	3
1. Overview	3
1.1. Executive Summary	4
1.2. Goals of the Assessment	4
1.3. Non-goals and Limitations	4
1.4. Results	4
2. Introduction	5
2.1. About Agora Cross-Chain (LayerZero)	6
2.2. Methodology	6
2.3. Scope	8
2.4. Project Overview	8
2.5. Project Timeline	9
3. Detailed Findings	9
3.1. Cross-chain receiving is not blocked when the contract is paused	10
3.2. Test coverage	11
4. System Design	12
4.1. Component: AgoraMintBurnOFTAdapter	13
5. Assessment Results	15
5.1. Disclaimer	16

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the [#1 CTF \(competitive hacking\) team](#) worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io and follow [@zellic_io](#) on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io.



1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for the Agora team from October 13th to October 22nd, 2025. During this engagement, Zellic reviewed Agora Cross-Chain (LayerZero)'s code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Has the upgradability been implemented correctly?
 - Is access control implemented correctly and not bypassable?
 - Is the inbound/outbound rate limits applied correctly?
 - Is it possible to bypass the rate limits?
-

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody

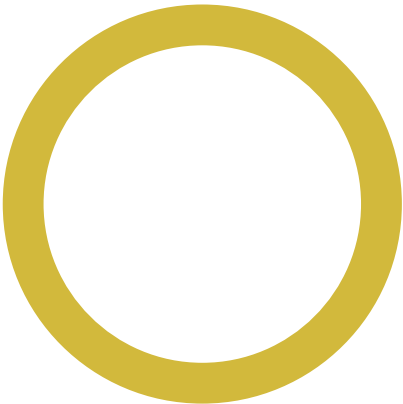
Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. Results

During our assessment on the scoped Agora Cross-Chain (LayerZero) contracts, we discovered two findings, both of which were medium impact.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	2
Low	0
Informational	0



2. Introduction

2.1. About Agora Cross-Chain (LayerZero)

The Agora team contributed the following description of Agora Cross-Chain (LayerZero):

Upgradeable LayerZero mint-and-burn OFT Adapter for Agora Dollar and access control layer.

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and

Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

2.3. Scope

The engagement involved a review of the following targets:

Agora Cross-Chain (LayerZero) Contracts

Type	Solidity
Platform	EVM-compatible
Target	layerzero-dev
Repository	https://github.com/amphora-atlas/layerzero-dev ↗
Version	b501b34a1b7fb638782c0804d6302ea01ffdb259
Programs	AgoraMintBurnOFTAdapter AgoraOFTAdapterAccessControl AgoraRateLimiter LZRateLimiterUpgradeable

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of 2.3 person-weeks. The assessment was conducted by two consultants over the course of one and a half calendar weeks.

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
✈ Engagement Manager
jacob@zellic.io ↗

Chad McDonald
✈ Engagement Manager
chad@zellic.io ↗

Pedro Moura
✈ Engagement Manager
pedro@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Katerina Belotskaia
✈ Engineer
kate@zellic.io ↗

Jaeeu Kim
✈ Engineer
jaeeu@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

October 13, 2025 Start of primary review period

October 14, 2025 Kick-off call

October 22, 2025 End of primary review period

3. Detailed Findings

3.1. Cross-chain receiving is not blocked when the contract is paused

Target	AgoraMintBurnOFTAdapter		
Category	Coding Mistakes	Severity	Medium
Likelihood	Medium	Impact	Medium

Description

The AgoraMintBurnOFTAdapter supports pausable functionality, which allows to pause the contract in the event of a emergency. The `_debit` function contains the `_requireNotPaused` check to not allow outbound bridging if the contract is paused. However, the `_credit` function does not contain this kind of verification. As a result, the cross-chain-receiving functionality is not blocked when the contract is paused.

Impact

Pausing currently only stops outbound sending; incoming cross-chain receiving is still allowed. If pausing is meant for a full emergency freeze, incoming cross-chain receiving must also be restricted during the pause period.

Recommendations

We recommend implementing the following change to the `_credit` function:

```
function _credit(
    address _to,
    uint256 _amountLocalDecimals,
    uint32 // _sourceEndpointId
) internal virtual override returns (uint256 amountReceivedLocalDecimals) {
    _requireNotPaused();
    // [...]
}
```

Remediation

This issue has been acknowledged by Agora team, and a fix was implemented in commit [b501b34a](#).

3.2. Test coverage

Target	Project Wide		
Category	Protocol Risks	Severity	Medium
Likelihood	N/A	Impact	Medium

Description

When building a complex contract with multiple moving parts (state changes) and dependencies, comprehensive testing is essential. This includes testing for both positive and negative scenarios on every function that touches the contract's state. Positive tests should verify that each function's side effect is as expected, while negative tests should cover every revert, preferably in every logical branch.

Despite extensive test coverage, some edge cases are not yet covered.

Impact

Good test coverage has multiple effects:

- It finds bugs and design flaws early (preaudit or prerelease).
- It gives insight into areas for optimization (e.g., gas cost).
- It displays code maturity.
- It bolsters customer trust in the product.
- It improves understanding of how the code functions, integrates, and operates — for developers and auditors alike.
- It increases development velocity long-term.

The last point may seem contradictory given the time investment to create and maintain tests. To expand upon it, tests help developers trust their own changes. It is difficult to know if a code refactor — or even just a small one-line fix — breaks other code if there are no tests. This is especially true for new developers or those returning to the code after a prolonged absence.

Tests have your back here. They are an excellent indicator that the existing functionality was most likely not broken by a change to the code. Without a comprehensive test suite, the above benefits are lost. This increases the likelihood of bugs and vulnerabilities going unnoticed until after deployment, which can be costly and damaging to the project's reputation.

Recommendations

We recommend to implement the test cases provided below.

- An account without `RATE_LIMIT_MANAGER_ROLE` cannot revoke the `RATE_LIMIT_EXEMPT_ROLE` from a set of addresses.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot revoke the `PAUSER_ROLE` from another account.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot revoke the `RATE_LIMIT_MANAGER_ROLE` from another account.
- The limit rate is disabled when the window is 0.

Remediation

This issue has been acknowledged by Agora team, and a fix was implemented in commit [7a4e18dc](#).

4. System Design

This provides a description of the high-level components of the system and how they interact, including details like a function's externally controllable inputs and how an attacker could leverage each input to cause harm or which invariants or constraints of the system are critical and must always be upheld.

Not all components in the audit scope may have been modeled. The absence of a component in this section does not necessarily suggest that it is safe.

4.1. Component: AgoraMintBurnOFTAdapter

Description

The AgoraMintBurnOFTAdapter implements the LayerZero OFTAdapterUpgradeable functionality, which utilizes the ERC-20 token's external mint and burn logic to enable cross-chain transfers. This contract is designed to be used in conjunction with the AgoraRateLimiter contract to enforce rate limits on cross-chain transferring and receiving tokens. Also, it inherits from the OFTAdapterUpgradeable, AgoraOFTAdapterAccessControl, PausableUpgradeable, and Erc1967Implementation contracts.

AgoraOFTAdapterAccessControl is used to manage the access control for the contract, which includes the PAUSER_ROLE, ACCESS_CONTROL_MANAGER_ROLE, RATE_LIMIT_MANAGER_ROLE, and RATE_LIMIT_EXEMPT_ROLE:

- The PAUSER_ROLE is responsible for pausing and unpausing the contract using the `pause()` and `unpause()` functions.
- The ACCESS_CONTROL_MANAGER_ROLE can grant and revoke the PAUSER_ROLE and the RATE_LIMIT_MANAGER_ROLE.
- The RATE_LIMIT_MANAGER_ROLE can grant and revoke the RATE_LIMIT_EXEMPT_ROLE to a set of addresses.
- Accounts with RATE_LIMIT_EXEMPT_ROLE are allowed to bypass rate limit verification for sending and receiving cross-chain tokens.

Additionally, the Erc1967Implementation contract is used to implement the ERC-1967 proxy pattern for the contract.

Invariants

- When paused, the cross-chain transferring functionality is blocked. Receiving tokens is still allowed.
- Only an account with PAUSER_ROLE can pause and unpause the contract.
- Frozen accounts cannot send tokens cross-chain, but frozen accounts are allowed to receive tokens.
- A contract cannot be initialized more than once.
- The accounts with RATE_LIMIT_EXEMPT_ROLE do not affect the current `amountInFlight` and `lastUpdated` state during cross-chain sending or receiving tokens.

- The current `amountInFlight` never exceeds the rate limit for output and input bridging.
- Sending cross-chain tokens will burn the `amountSentLocalDecimals` amount returned by the `_debitView` function — and it cannot be less than the `minAmountLocalDecimals` amount.
- The `lastUpdated` is monotonic nondecreasing.
- The received message with a GUID in the `exemptGuids` list can skip the rate-limit check.
- When the window is 0, the configured notional amount is equal the maximum that can be sent per second.
- Sending reverts if the destination EID is not supported.
- Only a trusted endpoint address can initiate `lzReceive` function execution.

Test coverage

Cases covered

- Sending tokens should reduce the caller balance by the `amountSentLocalDecimals` amount.
- Sending tokens does not change the contract balance.
- Receiving tokens should increase the recipient balance by the `_amountLocalDecimals` amount.
- The compose message can be successfully sent and received by the recipient.
- The `approvalRequired` is false.
- The compose message is decoded correctly.
- The debit function reverts if `_minAmountLocalDecimals` is less than the `amountSentLocalDecimals` amount.
- The `proxyAdminAddress` is equal to the expected `agoraDollarProxyAdmin` address.
- The `implementationAddress` is equal to the expected `adapterImplementationAddress`.
- An account with `ACCESS_CONTROL_MANAGER_ROLE` cannot revoke their own role.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot grant the `ACCESS_CONTROL_MANAGER_ROLE` to another account.
- The `grantAccessControlManagerRole` function emits the expected event.
- The `revokeAccessControlManagerRole` function emits the expected event.
- An account with `ACCESS_CONTROL_MANAGER_ROLE` can grant and revoke the `PAUSER_ROLE` to another account.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot grant the `PAUSER_ROLE` to another account.
- An account with `PAUSER_ROLE` can pause and unpause the contract.
- An account without `PAUSER_ROLE` cannot pause and unpause the contract.
- An account with `ACCESS_CONTROL_MANAGER_ROLE` can grant and revoke the `RATE_LIMIT_MANAGER_ROLE` to another account.
- An account with `RATE_LIMIT_MANAGER_ROLE` can set the rate limit.
- An account without `RATE_LIMIT_MANAGER_ROLE` cannot set the rate limit.

- An account without `RATE_LIMIT_MANAGER_ROLE` cannot execute the `setRateLimitExemptGuids` function.
- The `setRateLimitExemptGuids` function emits the expected event.
- The `setRateLimits` function emits the expected event.
- An account with `RATE_LIMIT_MANAGER_ROLE` can grant and revoke the `RATE_LIMIT_EXEMPT_ROLE` to a set of addresses.
- An account without `RATE_LIMIT_MANAGER_ROLE` cannot grant the `RATE_LIMIT_EXEMPT_ROLE` to a set of addresses.
- The `getAccessControlManagerRoleMembers`, `getPauserRoleMembers`, `getRateLimitExemptRoleMembers`, and `getRateLimitManagerRoleMembers` functions return the expected addresses.
- If the limit is exceeded, the `send` function reverts.
- Cross-chain sending will be performed successfully if the limit is not reached.
- A user with `RATE_LIMIT_EXEMPT_ROLE` can send above the rate limit.
- A receiver with `RATE_LIMIT_EXEMPT_ROLE` can receive tokens above the rate limit.
- If the received message with a GUID is in the `exemptGuids` list, the rate-limit check is skipped.
- A reached rate limit resets after the window period.
- Zero-amount sending reverts.
- Tokens cannot be sent when the adapter is paused.
- Different destinations share the same rate limit.

Cases not covered

- An account without `RATE_LIMIT_MANAGER_ROLE` cannot revoke the `RATE_LIMIT_EXEMPT_ROLE` from a set of addresses.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot revoke the `PAUSER_ROLE` from another account.
- An account without `ACCESS_CONTROL_MANAGER_ROLE` cannot revoke the `RATE_LIMIT_MANAGER_ROLE` from another account.
- The limit rate is disabled when the window is 0.

5. Assessment Results

During our assessment on the scoped Agora Cross-Chain (LayerZero) contracts, we discovered two findings, both of which were medium impact.

5.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.