Zellic

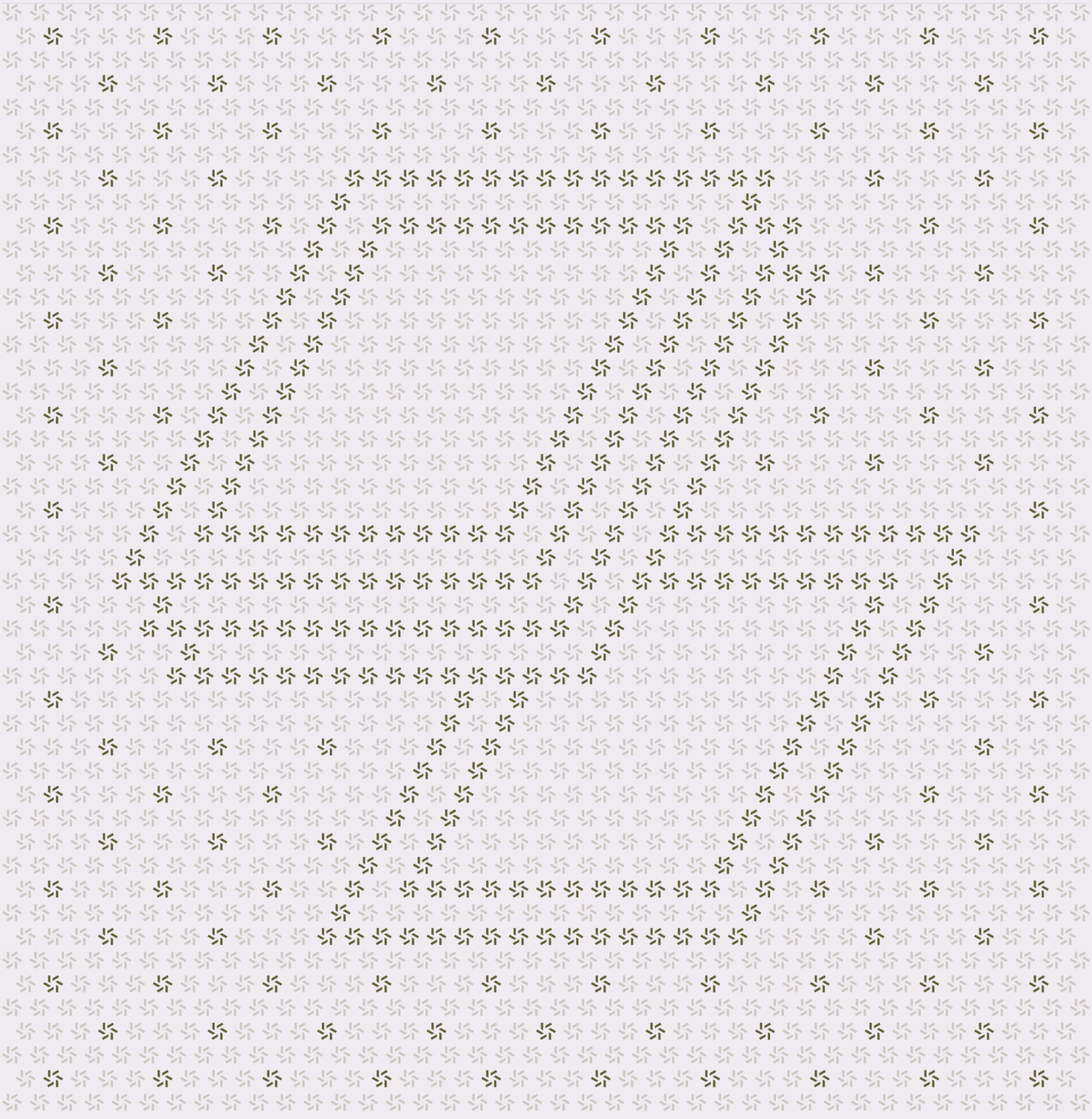
Prepared for
Eric Nordelo
OpenZeppelin

Prepared by
Jinseo Kim
Jisub Kim
Zellic

November 11, 2025

OpenZeppelin Cairo Contracts

Cairo Application Security Assessment



Contents

About Zellic **4**

1. Overview **4**

1.1. Executive Summary 5

1.2. Goals of the Assessment 5

1.3. Non-goals and Limitations 5

1.4. Results 5

2. Introduction **6**

2.1. About OpenZeppelin Cairo Contracts 7

2.2. Methodology 7

2.3. Scope 9

2.4. Project Overview 11

2.5. Project Timeline 11

3. Detailed Findings **11**

3.1. Foolproof logic on changing the delay for default admin transfer is incomplete 12

3.2. The `ERC6372Clock::clock()` must fit in `u48` range 14

3.3. Inconsistent state transition logic with Solidity implementation 15

3.4. The `Errors::NEGATIVE_FEE` constant is unused 17

3.5. Value of shares may be incorrect in the hook when fee is charged 18

3.6. Block-number clocks should require a block for delay 19

4. Discussion **19**

4.1. Notable changes 20

4.2. TimelockController has not yet integrated the AccessControlDefaultAdminRules 21

5. Assessment Results **21**

5.1. Disclaimer 22

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 OpenZeppelin from October 27th to November 3rd, 2025. During this engagement, Zellic reviewed the OpenZeppelin Cairo contracts' 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:

- Does the library adhere to robust and good coding practices?
- Does AccessControlDefaultAdminRules match the behavior of the Solidity implementation?
- Does the ERC-4626 fee trait work as expected?
- For ERC-4626, are there any specific scenarios where the vault might break or behave unexpectedly that require additional validation or documentation?

1.3. Non-goals and Limitations

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

- Optimizing runtime performance
- Infrastructure relating to the project

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

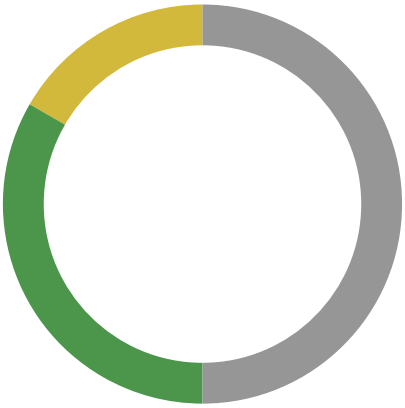
1.4. Results

During our assessment on the scoped OpenZeppelin Cairo contracts, we discovered five findings. No critical issues were found. One finding was of medium impact, two were of low impact, and the remaining findings were informational in nature.

Additionally, Zellic recorded its notes and observations from the assessment for the benefit of OpenZeppelin in the Discussion section (4. ↗).

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	1
Low	2
Informational	3



2. Introduction

2.1. About OpenZeppelin Cairo Contracts

OpenZeppelin contributed the following description of the OpenZeppelin Cairo contracts:

> A library for secure smart contract development written in Cairo for Starknet, a decentralized ZK Rollup. The library contains a set of components, contract presets, and utilities intended as backbone for different Starknet protocols implemented in Cairo.

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 targets.

Architecture risks. This encompasses potential hazards originating from the blueprint of a system, which involves its core validation mechanism and other architecturally significant constituents influencing the system's fundamental security attributes, presumptions, trust mode, and design.

Arithmetic issues. This includes but is not limited to integer overflows and underflows, floating-point associativity issues, loss of precision, and unfavorable integer rounding.

Implementation risks. This encompasses risks linked to translating a system's specification into practical code. Constructing a custom system involves developing intricate on-chain and off-chain elements while accommodating the idiosyncrasies and challenges presented by distinct programming languages, frameworks, and execution environments.

Availability. Denial-of-service attacks are another leading issue in custom systems. Issues including but not limited to unhandled panics, unbounded computations, and incorrect error handling can potentially lead to consensus failures.

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.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped targets itself. These observations — found in the Discussion (4. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved a review of the following targets:

OpenZeppelin Cairo Contracts Targets

Type	Cairo
Platform	Starknet

Target	cairo-contracts
Repository	https://github.com/OpenZeppelin/cairo-contracts ↗
Version	`Only the changes between f1c15a3...321064b`

Programs	`packages/governance/src/governor/extensions/governor_votes.cairo`
	`packages/governance/src/governor/extensions/governor_votes_quorum_fraction.cairo`
	`packages/governance/src/governor/governor.cairo`
	`packages/governance/src/timelock/timelock_controller.cairo`
	`packages/governance/src/votes/votes.cairo`
	`packages/utils/src/contract_clock.cairo`
	`packages/utils/src/contract_clock/block_number.cairo`
	`packages/utils/src/contract_clock/erc_6372.cairo`
	`packages/utils/src/contract_clock/timestamp.cairo`
	`packages/utils/src/lib.cairo`
	`packages/access/src/accesscontrol.cairo`
	`packages/access/src/accesscontrol/accesscontrol.cairo`
	`packages/access/src/accesscontrol/extensions.cairo`
	`packages/access/src/accesscontrol/extensions/accesscontrol_default_admin_rules.cairo`
	`packages/access/src/accesscontrol/extensions/pending_delay.cairo`
	`packages/token/src/erc20/extensions/erc4626/erc4626.cairo`
	`packages/account/src/extensions/src9/src9.cairo`
	`packages/account/src/utils.cairo`
	`packages/token/src/erc20/erc20.cairo`
	`packages/token/src/erc20/extensions/erc4626.cairo`
	
	`packages/access/src/accesscontrol/extensions/accesscontrol_default_admin_rules.cairo`
	`packages/token/src/common/erc2981/erc2981.cairo`
	`packages/presets/src/lib.cairo`
	`packages/presets/src/meta_tx_v0.cairo`
	`packages/account/src/account.cairo`
	`packages/account/src/eth_account.cairo`
	`packages/utils/src/execution.cairo`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of nine person-days. The assessment was conducted by two consultants over the course of six calendar days.

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:

Jinseo Kim
Engineer
jinseo@zellic.io

Jisub Kim
Engineer
jisub@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

October 27, 2025	Start of primary review period
October 28, 2025	Kick-off call
November 3, 2025	End of primary review period

3. Detailed Findings

3.1. Foolproof logic on changing the delay for default admin transfer is incomplete

Target	AccessControlDefaultAdminRules		
Category	Business Logic	**Severity**	Medium
Likelihood	Medium	**Impact**	Medium

Description

The AccessControlDefaultAdminRules requires a wait period every time the default admin-transfer delay changes. When the new delay is greater than the current one, the wait period is `min(new_delay, 5 days)`, and when the new delay is less than the current one, the wait period is `current_delay - new_delay`.

The comment ↗ says this five-day cap is meant to fix mistakes such as using milliseconds instead of seconds:

```
// When increasing the delay, we schedule the delay change to occur after a
    period of
// "new delay" has passed, up to a maximum given by
    defaultAdminDelayIncreaseWait, by
// default 5 days. For example, if increasing from 1 day to 3 days, the new
    delay will
// come into effect after 3 days. If increasing from 1 day to 10 days, the new
    delay
// will come into effect after 5 days. The 5 day wait period is intended to be
    able to
// fix an error like using milliseconds instead of seconds.
```

However, the current logic does not actually resolve the described issue.

Assume one tries to change the delay from one day to two days. When changing the delay, there are largely four possible mistake paths:

1. The previous delay was one day → the new delay would be 2,000 days.
2. The previous delay was one day → the new delay would be 0.002 days.
3. The previous delay was 1,000 days → the new delay would be two days. (Note that this is possible when a contract is initialized with an incorrect delay.)
4. The previous delay was 0.001 days → the new delay would be two days.

The five-day cap only affects the first case, and in our opinion, its effect is not desirable.

Without this cap, after the mistake on case 1, it would take 2,000 days for the delay to be updated, so one could notice the error and cancel the change. With this cap, the change is finalized after five days — since then it would take at least 2,000 days to transfer the default admin, and it would be impossible to effectively shorten this delay due to the way the wait period is calculated.

Impact

Unlike what the comment explains, the logic calculating the wait period to change the default admin transfer delay does not actually prevent user mistakes. On the contrary, it helps the unreasonably high delay to finalize quickly.

Recommendations

Consider enforcing explicit minimum and maximum bounds for `default_admin_delay`, limit how much the delay can change in one step, and keep the five-day cap only for normal short-term increases so mistakes cannot lock the system.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1567 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

3.2. The `ERC6372Clock::clock()` must fit in `u48` range

Target	ERC6372Clock		
Category	Integration Risks	**Severity**	Low
Likelihood	Low	**Impact**	Low

Description

Even though `ERC6372Clock::clock()` returns `u64`, external integrations aligned with EIP-6372 expect `uint48` values. The current `ERC6372TimestampClock` and `ERC6372BlockNumberClock` implementations do not document or enforce guards against values that exceed the `u48` range.

Impact

External integrations that receive or compute timepoints outside the `u48` range may hit errors that block proposal flows.

Recommendations

Consider adding documentation and interface comments to state that `clock()` must stay within the `u48` range, and include overflow checks in examples or reference implementations to guide integrators.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1600 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

3.3. Inconsistent state transition logic with Solidity implementation

Target	Governor		
Category	Business Logic	**Severity**	Low
Likelihood	Low	**Impact**	Low

Description

The proposal state transition logic in Cairo differs from the Solidity implementation when `voting_delay` is set to `0`. In Solidity, the condition for returning `Pending` state is:

```
if (snapshot >= currentTimepoint) {
    return ProposalState.Pending;
}
```

In Cairo, the condition is:

```
if current_timepoint < snapshot {
    return ProposalState::Pending;
}
```

The difference between `>=` and `<` means that when `voting_delay` is `0` and the snapshot equals the current block `N`:

- **Solidity**: Returns `Pending`, and voting attempts are rejected at state validation with a clear error indicating the proposal is not yet active.
- **Cairo**: Returns `Active`, passes state validation, but then reverts at `get_past_votes` with `Votes: future Lookup` error.

Impact

In both implementations, voting succeeds starting from `snapshot + 1`. However, the inconsistency causes different error messages when attempting to vote at the snapshot block. This misalignment with the Solidity implementation results in a less clear error message when voting fails at the snapshot block.

Recommendations

Consider aligning the state transition logic with the Solidity implementation by changing the condition to `current_timepoint <= snapshot` for returning `Pending` state.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1606 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

3.4. The `Errors::NEGATIVE_FEE` constant is unused

Target	ERC4626Component		
Category	Code Maturity	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

The `Errors::NEGATIVE_FEE` constant is defined, but nothing in the ERC-4626 flow calls it. Moreover, `FeeConfigTrait` hooks only return `Fee::Assets` or `Fee::Shares`, and both store values in `u256`, so they cannot hold a negative fee.

Impact

This may lead to confusion that there is logic handling negative fees.

Recommendations

Consider removing the unused `Errors::NEGATIVE_FEE` constant.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1599 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

3.5. Value of shares may be incorrect in the hook when fee is charged

Target	ERC4626Component		
Category	Code Maturity	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

When a fee is deducted on the ERC-4626 contract, a developer is responsible to implement the fee-deduction logic in the hook. Simultaneously, the developer might want to add certain business logic on the same hook.

The value of a share is expected to be mostly unchanged with deposit/withdraw/mint/redeem operations. But *during* the operation, including the fee-charging logic that should be manually implemented by developers, this invariant does not hold.

Impact

Some fee-charging cases require the assets to be transferred out from the vault to the fee recipient (in the `before_withdraw` and `after_deposit` hooks) or the shares to be minted to the fee recipient (in the `after_deposit` hook). As a result, before the fee-charging logic in the `after_deposit` hook or after the fee-charging logic in the `before_withdraw` hook, the value of a share would be incorrect.

In this period, a developer should be careful when the contract makes external calls (including deposit/withdrawing on the ERC-4626 contract itself) or tries to calculate the value of a share as it can be incorrect.

Recommendations

The current codebase, as of the time of writing, warns making an external call in the hook, but it does not mention that fee deduction is part of the deposit/withdraw operation and the value of a share can be incorrect when the operation is performed incompletely. Consider mentioning this caveat in a comment and/or documentation.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1602 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

3.6. Block-number clocks should require a block for delay

Target	Votes		
Category	Business Logic	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

The functions `Votes::get_past_votes` and `Votes::get_past_total_supply` require `timepoint < Clock::clock()`. When the governor uses a block-number clock and `voting_delay` equals `0`, the snapshot block equals the current block. Any vote cast in that same block reverts with `Votes: future lookup`, so votes only succeed starting from the next block.

Impact

This behavior is consistent with the Solidity implementation, where voting also succeeds starting from `snapshot + 1`. It is not a bug but an expected behavior that developers and users should be aware of when configuring a Governor with a block-number clock and zero voting delay.

Recommendations

Consider extending the documentation to clarify that when using block-number clocks with `voting_delay == 0`, votes can only be cast starting from the block after the proposal snapshot.

Remediation

This issue has been acknowledged by OpenZeppelin, and a fix was implemented in PR #1598 ↗. They merged these changes into the v3.0.0-rc branch accordingly.

4. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

4.1. Notable changes

This section outlines notable changes within several components.

Component 1: 3.0.0-alpha.0 (2025-07-18)

Here are notable changes in commits `1b67e094efce87fdd697673b1dfe4238d39163ef`, `cbfda3c12f82888b2108c480ab9a71e625dc9743`, `8eaed3eff1f4db5645632d51587ff782a49cf449`, and `ddaf32905714687d74cc650a988fbb78099f1d2f`.

- The ERC6372Clock interface under `openzeppelin_utils::contract_clock` was introduced. The `ERC6372Clock::clock()` must fit in `uint48` range; please see Finding 3.2. ↗ for more details.
- The AccessControlDefaultAdminRules interface and component were added. An issue was found within the cap increasing the admin delay; please see Finding 3.1. ↗ for more details.
- GovernorComponent and extensions were updated to support voting tokens that follow the ERC-6372 clock standard.
- VotesComponent was updated to allow configurable clock mechanisms via ERC6372Clock (breaking change). When `voting_delay` is set to zero, the state transition logic differs slightly from the Solidity implementation; please see Finding 3.3. ↗ for more details.

Component 2: 3.0.0-alpha.1 (2025-08-18)

Here are notable changes in commits `6cba8d01c53a665a0255664e7b4bb0ccdd76e50a` and `7739c6c451b9a7cbe9fb387effa69709ad1859ad`.

- The `AssetsManagementTrait` that defines asset-management strategies for the ERC4626Component was added.
- The `openzeppelin_interfaces` package, containing interfaces, ABIs, and dispatchers, was introduced.
- The scarb was bumped to version 2.12.0.
- ERC4626Component was enabled to integrate alternative asset-management strategies (including external vaults) via `AssetsManagementTrait`.
- FeeConfigTrait was refactored so that fees can be charged in either assets or shares.
- Extra input parameters were added to the ERC4626HooksTrait functions.

- Interfaces, ABIs, dispatchers, and related structs and types were moved into `openzeppelin_interfaces`.

Component 3: 3.0.0-alpha.2 (2025-09-10)

Here are notable changes in commits `5f821d7ece8de614acde7e6dea18740d28943ec6` and `eac5743d60bbd205872d48946b911a1f84f695f0`.

- The MetaTransactionV0 preset, interface, and dispatchers were added.
- The embeddable ERC2981AdminAccessControlDefaultAdminRulesImpl was added for managing ERC-2981 tokens via AccessControlDefaultAdminRules. TimelockController has not yet integrated the AccessControlDefaultAdminRules in this upgrade; OpenZeppelin plans to add it in a future update. Please see Discussion point 4.2. ↗ for more details.

Component 4: PR #1531

Here is a notable change in commits `d55344a57facdfe43847b75c8d4a4d1a5f4d4d5e`, `fee918625b776dcf7aa51933a6503c323bef275b`, `22d1aca10edef459ba2f084a2e236c0ea2a9294e`, `f9b8b26e6f43f75b9a15226f11f71a1e872f2508`, `306d2042ba7e1e26a4ebb50d8fafeef9cd775270`, and `dcc065cd5e0806211ee21cb0e0c0eb231c72a6b8`.

- Selected utilities were decoupled from the main package to allow independent versioning.

4.2. TimelockController has not yet integrated the AccessControlDefaultAdminRules

At the time of writing, the library includes two components that provide admin functionality on top of AccessControl: ERC-2981 and TimelockController. Commit `eac5743d` ↗ introduced the new AccessControlDefaultAdminRules extension, and its admin functions are already wired up for ERC-2981, so deployers can opt into that configuration at this moment.

However, the same extension has not yet been integrated into TimelockController because the work is still pending; OpenZeppelin plans to add it in a future update.

5. Assessment Results

During our assessment on the scoped OpenZeppelin Cairo contracts, we discovered five findings. No critical issues were found. One finding was of medium impact, two were of low impact, and the remaining findings were informational in nature.

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.