yAudit

November 25, 2025

Prepared for
Euler securitize

Audited by
Panda
HHK

Euler securitize

Smart Contract Security Assessment

Contents

1 Review Summary

1.1 Protocol Overview

ERC4626EVCCollateralSecuritize is an EVC-compatible collateral vault designed for Securitize RWA tokens, enforcing compliance-checked transfers through the Securitize compliance service while providing governance-controlled freeze and seize capabilities for regulated asset management.

1.2 Audit Scope

This audit covers eight smart contracts across two days of review.

```
src/
├── Swaps/
│   ├── SwapVerifier.sol
│   └── TransferFromSender.sol
├── Vault/
│   ├── implementation/
│   │   ├── ERC4626EVC.sol
│   │   ├── ERC4626EVCCollateral.sol
│   │   ├── ERC4626EVCCollateralCapped.sol
│   │   └── ERC4626EVCCollateralFreezable.sol
│   └── deployed/
│       └── ERC4626EVCCollateralSecuritize.sol
└── VaultFactory/
    └── ERC4626EVCCollateralSecuritizeFactory.sol
```

1.3 Risk Assessment Framework

1.3.1 Severity Classification

1.4 Key Findings

Breakdown of Finding Impacts

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

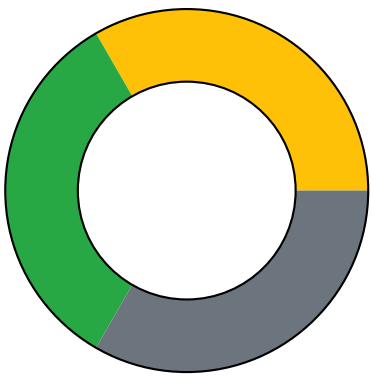

Figure 1: Distribution of security findings by impact level

Severity	Description	Potential Impact
Critical	Immediate threat to user funds or protocol integrity	Direct loss of funds, protocol compromise
High	Significant security risk requiring urgent attention	Potential fund loss, major functionality disruption
Medium	Important issue that should be addressed	Limited fund risk, functionality concerns
Low	Minor issue with minimal impact	Best practice violations, minor inefficiencies
Undetermined	Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and although their exact consequences remain uncertain, they present a sufficient potential risk to warrant attention and remediation.	Varies based on actual severity
Gas	Findings that can improve the gas efficiency of the contracts.	Reduced transaction costs
Informational	Code quality and best practice recommendations	Improved maintainability and readability

Table 1: severity classification

1.5 Overall Assessment

The codebase demonstrates strong security practices with no critical or high-severity vulnerabilities identified. The medium-severity finding regarding frozen account borrowing behavior has a clear mitigation path, while the low and informational findings represent edge cases and design considerations rather than fundamental flaws. The code quality is solid and production-ready with the recommended fixes applied.

2 Audit Overview

2.1 Project Information

Protocol Name: Euler securitize
Repository: https://github.com/euler-xyz/evk-periphery/
Commit Hashes:

- 101860e5a2b6cdf469f0888faa3cad67e6236c95
- 6d5a5af16773b26ff62828cab0305e9809958855

Commit URLs:

- https://github.com/euler-xyz/evk-periphery/blob/101860e5a2b6cdf469f0888faa3cad67e6236c95/

- https://github.com/euler-xyz/evk-periphery/blob/6d5a5af16773b26ff62828cab0305e9809958855/

2.2 Audit Team

Panda, HHK

2.3 Audit Timeline

The audit was conducted from November 17 to 18, 2025.

2.4 Audit Resources

Code repositories, documentation, previous audit

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

2.7.1 Frozen accounts can still borrow against frozen collateral

When the governor freezes an account, the collateral becomes unavailable and cannot be liquidated. However, the current architecture still allows the account to borrow tokens, potentially bypassing the freeze's intended purpose.

Technical Details

The function `freeze()` allows freezing the assets of a user. This freeze functionality is required by the Securitize integration guide.
The freeze is intended to execute court orders or regulatory needs on an account. When frozen, an account cannot be liquidated, and eventually the governor will be responsible for unfreezing it or seizing the collateral.
However, the frozen account can still borrow on the EVK after being frozen because the EVK uses `balanceOf()` to determine the account balance and ignores if the assets are frozen.
This could allow a user under court order to max borrow against his collateral after being frozen and exit some liquidity, bypassing some of the intended purpose of the freeze feature.

Impact

Medium. A frozen account can still borrow against its collateral, potentially bypassing court orders or regulatory requests.

Recommendation

As suggested by the Euler team:
Override the `balanceOf()` function to return zero if the EVC checks are in progress. This will result in 0 collateral for the EVK and effectively block new borrows while still returning the correct user balance for wallets and indexers.

Developer Response

Fixed in: bb0bb5.

2.8 Low Findings

2.8.1 Frozen Accounts Cannot Add Collateral Before Unfreezing, Leading to Immediate Liquidation Risk

Technical Details

When an account is frozen via the `freeze()` function in `ERC4626EVCCollateralFreezable`, the `whenNotFrozen` modifier blocks all balance-changing operations including `deposit()`, `mint()`, `withdraw()`, `redeem()`, and `transfer()` operations for that account.
While the account owner can still repay outstanding debt on the borrowing vault side while frozen, they cannot add more collateral to this vault to improve their collateralization ratio. If during the freeze period the account becomes undercollateralized, the account will be immediately eligible for liquidation upon being unfrozen.
This creates an asymmetric situation where:

1. The frozen account cannot proactively add collateral to protect against liquidation
2. Upon unfreezing, liquidators can immediately liquidate the account if it's underwater
3. There is no grace period or opportunity for the account owner to add collateral between unfreezing and potential liquidation

Impact

Low.

Recommendation

Implement a grace period: Add a grace period mechanism after unfreezing during which liquidations are temporarily disabled, giving account owners time to add collateral if needed.

Developer Response

Acknowledged, will keep as is. Although this might be a valid issue for other freezable contracts, in the case of Securitize collateral, we prefer to keep the logic simple, as freezing will occur in extraordinary situations, which will be handled individually. Since the liquidators are whitelisted, the parties may arrange unfreezing as they see fit.

2.9 Gas Savings Findings

None.

2.10 Informational Findings

2.10.1 Immutable `complianceService` address cannot be updated

Technical Details

The `complianceService` address is a storage variable on the Securitize contracts and could be updated. The current implementation stores it in an immutable variable that cannot be updated.

Impact

Informational. If the `complianceService` address changes, the contracts will need to be redeployed.

Recommendation

Consider making `complianceService` a storage variable with a setter, or add a public function that queries the current `complianceService` from Securitize on demand.

Developer Response

Fixed in: 4b102cd.

2.11 Final Remarks

The codebase demonstrates strong security practices, no critical/high-severity issues, and only medium/low-severity findings that have clear mitigations and represent edge cases rather than fundamental flaws. However, some elements of the Securitize integration remain unclear and will require additional testing once those details are known.