Audit Boutique

Decentralized EURO (dEURO) Smart Contracts Audit Report

Produced for:
DFX AG

Produced by:
Dominik Spicher, Audit Boutique

January 13th 2025

Contents

1. Executive Summary

In December 2024 and January 2025, Audit Boutique was engaged by DFX to perform a smart contract audit of the Decentralized Euro (dEURO) stablecoin system, a fork of the FrankenCoin stablecoin. The audit has uncovered no issues of high severity, one issue of medium severity (fixed in the meantime), and one issue of low severity. We also list seven general suggestions of how to improve the smart contracts.

2. Provenance and Audit Scope

The Decentralized Euro is a fork of the FrankenCoin project,[1] a Swiss Franc stablecoin, which has been audited elsewhere.[2] The scope of this audit report thus only pertains to the code changes that have been made on top of the Frankencoin smart contract codebase.

2.1 Split from Frankencoin

The dEURO codebase is based on revision `fa1457b` of the FrankenCoin code.[3][4] Note that this corresponds precisely to the revised version of the code under review in the ChainSecurity audit of the FrankenCoin system.[5]

As of December 19th 2024, no relevant changes have been made since revision `fa1457b` to the FrankenCoin codebase.[6]

2.2 Audit Scope

The audit was performed on the smart contract source code files in the dEURO repository[7] on the following two versions:

- Version 1 (received on 2nd December 2024): `4a81cd2228b390b05b4cb46f8f84d52d10e71b7a`
- Version 2 (received on 15th December 2024): `6cb44e0d74c6cf7d16b9ac6f59c72322a10b6c6d`
- Version 3 (received on 19th December 2024): `fe9fa7404ad0a2d55c4baf13f50b0bd36f074af8`

The following smart contract source files were part of the audit:

- `DecentralizedEURO.sol`
- `Equity.sol`
- `Leadrate.sol`
- `Savings.sol`
- `StablecoinBridge.sol`
- `impl/ERC3009.sol`
- `MintingHubV2/MintingHub.sol`
- `MintingHubV2/PositionFactory.sol`
- `MintingHubV2/PositionRoller.sol`

---

[1] https://github.com/Frankencoin-ZCHF/FrankenCoin/
[2] https://github.com/Frankencoin-ZCHF/FrankenCoin/tree/main/audits
[3] See Appendix B.1
[4] Note that this is incorrectly stated in the `README.md` file of the dEURO project where it is claimed that revision `a2ce625` is the common ancestor: https://github.com/d-EURO/smartContracts/blob/fe9fa7404ad0a2d55c4baf13f50b0bd36f074af8/README.md#L25
[5] https://github.com/Frankencoin-ZCHF/FrankenCoin/blob/main/audits/V2/ChainSecurity_Frankencoin_Frankencoin_v2024.pdf
[6] See Appendix B.2
[7] https://github.com/d-EURO/smartContracts

- `MintingHubV2/Position.sol`
- `utils/DEPSWrapper.sol`
- `utils/MathUtil.sol`

See Appendix A for a list of file hashes under the respective reviewed versions.

Imported contracts from well-known libraries such as Open-Zeppelin were not part of the review.

The review was constrained to the Solidity source file. Low-level assembly code generated thereof was not inspected.

3. Methodology

The review consisted of the following steps:

- Analysis of previous audits
- Check for compliance with smart contract development best practices
- Check for compliance with specification, where available and applicable
- Manual inspection and analysis of the smart contract and test code
- Usage of static analysis tools

4. System description

We refrain from a full description of the stablecoin system introduced by the FrankenCoin project, which almost universally also applies to the dEURO project. Very briefly, one of its chief motivations and innovations is that it uses an auction mechanism to value posted collateral instead of requiring price oracles. For a more comprehensive description, see the ChainSecurity Code Assessment report,[8] section 2.2.

In the following, we instead describe the changes that have been by the Decentralized EURO project in comparison to its FrankenCoin base. We refrain from commenting on trivial changes such as renamings.

The system description pertains to the most recent version of the reviewed contracts (version 3).

4.1 Initial Equity Price Change

The initial price of the Equity token (called "native Decentralized Protocol Share" or "nDEPS") has been decreased thousandfold (ignoring the exchange rate difference) from 1000 ZCHF to 1 dEUR, and now applies to the first 10'000'000 Equity shares, instead of the initial 1'000 shares.

4.2 Removal of Lockup Period Before Interest Yield

In the FrankenCoin system, saved ZCHF are subject to a lockup period of up to three days before any interest is yielded.[9] In the dEURO system, this lockup period has been removed. As a consequence, there is no temporal constraint on when `Savings.withdraw()` can be called, and interest is yielded immediately.

4.3 Introduction of the Frontend Gateway System

The new Frontend Gateway consists of a) variants of the `Savings` and `MintingHub` contracts where a number of their functions are extended such that the owner of a particular frontend code receives a commission fee proportional to the involved amounts, as well as b) a dedicated `FrontendGateway` contract to manage and pay out the commissions. The `FrontendGateway` contract also contains a few entrypoints into the `Equity` contract where commissions are also accrued. Those two sets of functionalities experience different commission rates by default (5% and 1% respectively), although those rates can be changed by anybody who gathers the support of enough Equity holders. The commissions are covered by the reserve of the dEURO contract through its `coverLoss` function.

[8] https://github.com/Frankencoin-ZCHF/FrankenCoin/blob/b311b1069d21c421e0907ba292082fedfd6ea6ef/audits/V2/ChainSecurity_Frankencoin_Frankencoin_v2024.pdf

[9] https://github.com/Frankencoin-ZCHF/FrankenCoin/blob/b311b1069d21c421e0907ba292082fedfd6ea6ef/contracts/Savings.sol#L18-L21

4.4 ERC165 implementation

Both the Equity and DecentralizedEURO smart contracts now implement ERC165[10] and signal that they implement ERC20, ERC20Permit and ERC3009. Furthermore, the MintingHub and MintingHubGateway contracts now signal that they implement their respective interface definition.

4.5 ERC3009 implementation

The Equity and DecentralizedEURO smart contracts have inherited new ERC3009 functionality implemented by a new contract. This allows authorized transfers where the transaction fees can be paid by an address different from the transaction initiator. Because this is the only contract that uses a different `pragma solidity` specifier, we suspect that this source was copied from somewhere. Indeed, it contains large similarities to numerous deployed and / or published contracts, but its exact provenance could unfortunately not be determined.

4.6 Internal Exchange Fee Increase

The internal exchange fee (or "issuance fee"), charged when newly minted pool shares are issued to an investor, has been increased from 0.3% to 2%.

4.7 Stablecoin Bridge now supports differing numbers of decimals

When computing corresponding amounts for the dEURO coin on different chains (represented by another Ethereum ERC20 token), the `StablecoinBridge` contract now contains logic to handle differing numbers of decimals in the two tokens. This was previously not necessary for FrankenCoin because the only relevant bridged coin (XCHF) had the same number of decimals.

4.8 Usage of OpenZeppelin contracts

In the FrankenCoin system, custom implementations of standard token interfaces such as ERC20 have been used.[11] The Decentralized EURO project has replaced these with version 5 contracts of the well known OpenZeppelin smart contract suite.[12]

5. Best practices checklist

Non-adherence to the characteristics listed in the below list does not represent a security issue itself. However, following best practices is a good indicator of care and attention to detail. In addition, it allows the review process to be more focused and efficient, thus making it more likely for issues to be uncovered.

[10] https://eips.ethereum.org/EIPS/eip-165
[11] According to verbal testimony, they have been themselves been based on version 4 of OpenZeppelin smart contracts. Thus, appendix C analyzes the breaking changes in OpenZeppelin v5 according to their relevance to the dEURO smart contract suite.
[12] https://www.openzeppelin.com/solidity-contracts

- ✓ The code was provided as a Git repository
- ○ There exists specification, covering the most important aspects of smart contract functionality (only the issuance schedule is specified)
- ○ The development process is understandable through well-delineated, atomic commits
- ✓ Code duplication is minimal
- ✓ The smart contract source files are provided in an unflattened manner
- ✓ The code compiles with a recent Solidity version
- ✓ The code is consistently formatted
- ✓ Code comments are in line with the associated code
- ✓ There are tests
- ✓ The tests provide good coverage
- ✓ The code is well documented
- ✓ There is no commented code
- ✓ There is no unused code
- ○ The code follows standard Solidity naming conventions

6. Susceptibility to FrankenCoin Audit Findings

A full audit of the FrankenCoin code base was out of scope for this audit. However, we analyzed the two published audits of the FrankenCoin system and whether the dEURO codebase is susceptible to any of their published findings.

6.1 ChainSecurity Code Assessment

We analyze the code assessment published by ChainSecurity under the title "Code Assessment of the FrankenCoin v2024 Smart Contracts" on November 28, 2024.[13]

The code assessment groups security-relevant findings into two categories:

- Open findings (section 5)
- Resolved findings (section 6)

Note that sections 7 and 8 contain non-security relevant findings and general notes of interest respectively - both of which we will not analyze further due to their benign nature, but which may provide interesting insights nonetheless.

6.1.1 Open Security-Relevant Findings

Three low-severity findings have not been completely fixed as of the version which dEURO is based upon. For each, we briefly comment on further analysis the Decentralized EURO team may conduct to ensure that the dEURO smart contracts are not subject to unexpected behavior.

6.1.1.1 CS-ZCHF2-007: Frontrunning Denial of Service

The FrankenCoin team has chosen to accept a potential frontrunning Denial of Service attack on the `Savings.withdraw()` and `Position.repay()` functions. The dEURO team should ensure that they can accept this risk as well.

6.1.1.2 CS-ZCHF2-009: burnWithoutReserve Event Can Overestimate Profit

This issue remains acknowledged with a code comment in the dEURO codebase. However, we stress that the reasoning provided by the FrankenCoin team ("the Frankencoin contract is already deployed") does not apply, and it remains possible to fix this imprecision.

6.1.1.3 CS-ZCHF2-010: buyExpiredCollateral Can Leave Dust

The risk of stuck dust amounts was accepted by the FrankenCoin team. The dEURO team should ensure that this risk is acceptable for them as well.

---

13 https://github.com/Frankencoin-ZCHF/FrankenCoin/blob/b311b1069d21c421e0907ba292082fedfd6ea6ef/audits/V2/ChainSecurity_Frankencoin_Frankencoin_v2024.pdf

6.1.2 Resolved Security-Relevant Findings

All the findings listed in this section have been corrected in version 2 under review, which corresponds to the version which the Decentralized EURO project is based on,[14] thus inheriting all the fixes.

6.2 decurity Audit

We analyze the audit report published by decurity under the title “Smart Contract Security Audit Report - FrankenCoin Audit”.[15]

The audit report lists fifteen findings, ten of which are marked as fixed, and five of which are marked as acknowledged.

6.2.1 Acknowledged Findings

We briefly analyzed all (incompletely) fixed issues and their relevance to the dEURO smart contract system individually.

6.2.1.1 Finding 5.3

As stated in the finding description, this attack is both unlikely to pass the governance process, and risky for the attacker himself. The dEURO team should ensure that this is sufficient rationale for accepting the risk of a small `challengePeriod`.

6.2.1.2 Finding 5.4

The dEURO team should ensure that the stated rationale for accepting the risk in a depleted equity capital scenario applies to them as well.

6.2.1.3 Finding 5.6

Related to Finding 5.3 referenced in our section 6.2.1.1, this issue pertains to very low `challengePeriod` values. The dEURO team should ensure that they are in alignment with the stated FrankenCoin solution of “the resolution is to enforce reasonable values in the frontend.”

6.2.1.4 Finding 5.13

The dEURO team should ensure it accepts the risk of positions with a vanishing `riskPremiumPPM`.

6.2.1.5 Finding 5.14

The dEURO team should ensure it accepts the loss of precision in price calculations.

[14] See Appendix B.1

[15] https://github.com/Frankencoin-ZCHF/FrankenCoin/blob/b311b1069d21c421e0907ba292082fedfd6ea6ef/audits/V2/frankencoin-audit-report-2024-1.1.pdf

6.2.2 Fixed Findings

On top of the analysis performed in appendix B.2 (which establishes that all changes made to pertinent smart contracts that have been referenced by a commit must have been made before dEURO has been split off), we reconfirmed that the following referenced fixing commits are all part of the dEURO history:

- Finding 5.1: `502f786`
- Finding 5.2: `67cec26` and `3498f08`
- Finding 5.5: `f42227a`
- Finding 5.7: `c38c1c2`
- Finding 5.8: `6672676`
- Finding 5.9: `125a6c7`
- Finding 5.10: `d712153`
- Finding 5.11: `087f940`
- Finding 5.12: `103edb9`
- Finding 5.15: `7b2381a`

7. Findings

We rank our findings according to their perceived severity (high, medium, low), where an issue's severity is understood to be the product of its likelihood of being triggered and the impact of its consequences. Where deemed appropriate, we also report issues that are not security-relevant per-se but impact things like transaction-cost effectiveness or user experience.

See section 6.4 for suggestions for improvements which have no discernible impact on any of the above.

7.1 High severity

No issues found.

7.2 Medium severity

7.2.1 Failure to decrease allowance in when investing for another account

Status: Introduced in version 2, fixed in version 3.

When calling `investFor()` on the `Equity` contract, the contract checks whether the message sender holds a high enough allowance on the investor for this token contract. However, subsequently the allowance is not spent through the ERC20 `_spendAllowance` function, thus allowing the caller to re-use the same allowance for multiple calls.

7.3 Low severity

7.3.1 Large re-entrancy attack surface vulnerable to malicious collateral contracts[16]

Status: Introduced in the FrankenCoin system.

The `Position`, `PositionRoller` and `MintingHub` contracts contain numerous functions that make ERC20 calls to a-priori unknown collateral token smart contracts.[17] These calls exhibit a large re-entrancy attack surface, as listed in appendix D. However, the usage of a specific collateral token is subject to a tacit approval[18] of equity shareholders, which is why this issue is deemed low severity. Nevertheless, care should be taken to closely inspect proposed positions utilizing new token smart contracts.

[16] Note that this issue is also present in the FrankenCoin contracts.
[17] The comment on L569 of `Position.sol` suggests that accommodation should be made for exotic token smart contracts.
[18] In the sense that no veto is registered.

7.4 Informational

7.4.1 Unused helper function in MathUtil contract

Status: Introduced in version 1.

The `_min` helper function in the `MathUtil` contract is unused and can be removed.

7.4.2 Contract state variable could be declared immutable

Status: Introduced in the FrankenCoin system.

The `deuro` variable of the `PositionRoller` contract could be declared immutable to incur gas savings.

7.4.3 Unchecked return value from an external transfer

Status: Introduced in the FrankenCoin system.

The following functions ignore return values from calls to transfer or transferFrom functions of external contracts (collateral contracts). This is potentially dangerous because not all ERC20 tokens are known to revert upon unsuccessful transfers.

- `Position.adjust(uint256,uint256,uint256)` ignores `collateral.transferFrom(msg.sender,address(this),newCollateral - colbal)`
- `Position.withdraw(address,address,uint256)` ignores `IERC20(token).transfer(target,amount)`
- `Position._sendCollateral(address,uint256)` ignores `IERC20(collateral).transfer(target,amount)`
- `PositionRoller.roll(IPosition,uint256,uint256,IPosition,uint256,uint256,uint40)` ignores `targetCollateral.transferFrom(msg.sender,address(this),collDeposit)`
- `PositionRoller.roll(IPosition,uint256,uint256,IPosition,uint256,uint256,uint40)` ignores `targetCollateral.transferFrom(msg.sender,address(target),collDeposit)`
- `MintingHub.openPosition(address,uint256,uint256,uint256,uint40,uint40,uint40,uint24,uint256,uint24)` ignores `IERC20(_collateralAddress).transfer(address(0x123),invalidAmount)`
- `MintingHub.openPosition(address,uint256,uint256,uint256,uint40,uint40,uint40,uint24,uint256,uint24)` ignores `IERC20(_collateralAddress).transferFrom(msg.sender,address(pos),_initialCollateral)`
- `MintingHub.clone(address,address,uint256,uint256,uint40)` ignores `collateral.transferFrom(msg.sender,pos,_initialCollateral)`
- `MintingHub.challenge(address,uint256,uint256)` ignores `IERC20(position.collateral()).transferFrom(msg.sender,address(this),_collateralAmount)`

- `MintingHub._avertChallenge(MintingHub.Challenge,uint32,uint256,uint256)` ignores `_challenge.position.collateral().transfer(msg.sender,size)`
- `MintingHub.returnPostponedCollateral(address,address)` ignores `IERC20(collateral).transfer(target,amount)`
- `MintingHub._returnCollateral(IERC20,address,uint256,bool)` ignores `collateral.transfer(recipient,amount)`

7.4.4 Missing interface inheritance declaration

Status: Introduced in the FrankenCoin system.

The `PositionFactory` contract does not specify that it inherits from its interface `IPositionFactory`, thus foregoing benefits such as protection from accidental name shadowing or assured compliance with the interface definition.

Furthermore, the `PositionFactory` contract could also use the ERC165 mechanism employed elsewhere to signal that it implements its interface.[19]

7.4.5 Replicated logic from noCooldown modifier

Status: Introduced in the FrankenCoin system.

The `_withdrawCollateral()` function of the `Position` contract contains the identical check that the `noCooldown` modifier provides, and could thus use it instead.

7.4.6 Unnecessary return argument in notifyChallengeSucceeded function

Status: Introduced in the FrankenCoin system.

The last value in the returned tuple from the `notifyChallengeSucceeded` function of the `Position` contract is a public immutable variable of the contract, and can thus always be obtained independently, making it possible to simplify the function signature.

7.4.7 Incomplete test coverage

Status: Introduced in version 1.

Appendix F lists the test coverage for the DecentralizedEURO smart contract system.[20] Whereas the coverage is quite extensive in some places, other smart contract functionality, introduced or modified by the DecentralizedEURO system, remains untested:

- The `DEPSWrapper` system
- The Halley root finding procedure in `MathUtils`

[19] See section 4.4.
[20] Note that the tests themselves were not part of the audit.

- The change in the interest payment schedule introduced in the `Position` contract[21][22]

[21] See section 4.8.
[22] Despite the `Position` contract itself having excellent coverage, we were not able to ascertain based on the Git commit history that tests specific to the new interest payment schedule exist.

8. Limitations

Even though the code has been reviewed carefully on a best-effort basis, undiscovered issues can not be excluded. This report does not consist in a guarantee that no undeclared issues remain, nor should it be interpreted as such.

Appendix A: File Hashes

Due to the large number of involved versions, and for the avoidance of doubt for readers who only receive contract files, we list here the SHA256 prefixes for all files under review listed in section 2.2, for each of the reviewed versions listed in section 2.2.

Contract Source File	Version 1	Version 2	Version 3
DecentralizedEURO.sol	97738862ac977d7d	f0fff883797be885	f0fff883797be885
Equity.sol	58da6615317320ab	4a544f1c76899a0b	b17f2de8fa242529
Leadrate.sol	9510ebbafe6cc288	0ac78d01131d6e23	0ac78d01131d6e23
MintingHubV2/MintingHub.sol	08f9cf85d347c058	84d35fc2b61dd42f	84d35fc2b61dd42f
MintingHubV2/Position.sol	83fcbc06d0221830	3aa5d201ab9e0232	3aa5d201ab9e0232
MintingHubV2/PositionFactory.sol	390bc80e7523fadb	e5a93a040b765693	e5a93a040b765693
MintingHubV2/PositionRoller.sol	58fdbb84fb0f16a2	6174757a63438915	6174757a63438915
Savings.sol	8e177e0525496c89	1f7f5e7f59d653f6	1f7f5e7f59d653f6
StablecoinBridge.sol	7556f38033d89c48	55c30eccab81194e	55c30eccab81194e
impl/ERC3009.sol	c5410135b69ec5f0	c5410135b69ec5f0	c5410135b69ec5f0
utils/DEPSWrapper.sol	f4e6a00ab7166e45	f4e6a00ab7166e45	ec8a3fccdd8cf49a
utils/MathUtil.sol	06c01a5a519fb8b1	06c01a5a519fb8b1	06c01a5a519fb8b1

Appendix B: Code Provenance

For the reader's convenience, we document our analysis of the code provenance with respect to the FrankenCoin codebase.

The sections in this appendix all use the following Git remotes:

- `origin`: https://github.com/d-EURO/smartContracts
- `franken`: https://github.com/Frankencoin-ZCHF/FrankenCoin

Furthermore, we use the following branch pointers pointing to the specified revisions as of 19th December 2024:

- `origin/main`: `fe9fa7404ad0a2d55c4baf13f50b0bd36f074af8`
  - Note that this is Version 2 under review
- `franken/main`: `b311b1069d21c421e0907ba292082fedfd6ea6ef`

B.1 Split off of FrankenCoin

We determine the FrankenCoin version which the Decentralized EURO project is based on as follows:

```
$ git merge-base origin/main franken/main

fa1457b9c2b370fbcecc5442ae4e0c51b62e454f
```

B.2 Analysis of Changes to FrankenCoin Since the Split

In order to determine whether the Decentralized EURO project is missing out on relevant changes that have been made to the FrankenCoin project since the split, we observe the following:

```
$ git diff --name-only $(git merge-base origin/main franken/main) franken/main
'contracts/**/*.sol'

contracts/MintingHubV1/MintingHubV1.sol
contracts/MintingHubV1/PositionFactoryV1.sol
contracts/MintingHubV1/PositionV1.sol
contracts/MintingHubV1/interface/IPositionFactoryV1.sol
contracts/MintingHubV1/interface/IPositionV1.sol
```

And we conclude that the only changes that have been made to Solidity smart contract files are restricted to the version 1 of the MInting Hub, which is irrelevant to the Decentralized EURO project.

Appendix C: Breaking Changes in OpenZeppelin v5

As detailed in section 4.7, the FrankenCoin system had previously used homegrown implementations of standard token functionality, which are believed to be based on version 4 of OpenZeppelin, although this statement has not been verified rigorously. Nevertheless, in this appendix we briefly analyze the

communicated breaking changes in OpenZeppelin v5 and their relevance to the dEURO smart contracts, since the dEURO system replaced the custom implementations with v5 OpenZeppelin smart contracts.

The main property we are trying to uphold is that the FrankenCoin contracts did not rely on OpenZeppelin v4 behaviour[23] that was changed in version 5.

At the outset, we note that the OpenZeppelin project adheres[24] to semantic versioning,[25] which means it is enough to inspect the breaking changes for version 5.0.0.

Inspecting the changelog for version 5.0.0,[26] we notice a single relevant breaking change to the ERC20 token implementation where the previously overridden (by the `Equity` smart contract) `_beforeTokenTransfer()` function was removed in favour of a new overridable `_update()` function. This change has been made in accordance with the published migration guide.[27]

[23] Although, note again that the relevance of OpenZeppelin v4 to FrankenCoin remains conjectural.

[24] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/d4ed5f9068bb634ecc423417b4f9554f48fae85b/README.md#L19-L20

[25] https://semver.org/

[26] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/d4ed5f9068bb634ecc423417b4f9554f48fae85b/CHANGELOG.md#500-2023-10-05

[27] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/d4ed5f9068bb634ecc423417b4f9554f48fae85b/CHANGELOG.md#erc20-erc721-and-erc1155

Appendix D: Re-entrancy attack surface induced by calls to collateral contract

As detailed in section 7.3.1, we list here potential attack targets for collateral smart contracts re-entering into various functions of the `Position`, `PositionRoller` and `MintingHub` contracts.

D.1 State variables written after external calls

- Re-entrancy in `Position.adjust()`
  - Relevant external calls:
    - `collateral.transferFrom(msg.sender,address(this),newCollateral - colbal)`
    - `withdrawCollateral(msg.sender,colbal - newCollateral)`
      - `IERC20(collateral).transfer(target,amount)`
  - State variables written after the calls:
    - `_payDownDebt(msg.sender,debt - newDebt)`
      - `accruedInterest = debt - principal`
      - `accruedInterest -= interestToPay`
    - `_adjustPrice(newPrice)`
      - `cooldown = horizon`
- Re-entrancy in `Position.forceSale()`
  - Relevant external call:
    - `remainingCollateral = _sendCollateral(buyer,collAmount)`
      - `IERC20(collateral).transfer(target,amount)`
  - State variables written after the call:
    - `used = _payDownDebt(buyer,proceeds)`
      - `accruedInterest = debt - principal`
      - `accruedInterest -= interestToPay`
    - `_payDownDebt(buyer,deficit)`
      - `accruedInterest = debt - principal`
      - `accruedInterest -= interestToPay`

D.2 Events emitted after external calls

The following targets would only modify either the order or arguments of emitted events.

- Re-entrancy in `MintingHub.openPosition()`
  - Relevant external calls:
    - `IERC20(_collateralAddress).transfer(address(0x123),invalidAmount)`
    - `IERC20(_collateralAddress).transferFrom(msg.sender,address(pos),_initialCollateral)`
  - Event emitted after the calls:
    - `PositionOpened(msg.sender,address(pos),address(pos),_collateralAddress)`

- Re-entrancy in `MintingHub.bid()`
  - Relevant external call:
    - `_returnChallengerCollateral(_challenge,_challengeNumber,size,postponeCollateralReturn)`
      - `collateral.transfer(recipient,amount)`
  - Event emitted after the call:
    - `ChallengeSucceeded(address(_challenge.position),_challengeNumber,offer,transferredCollateral,size)`
- Re-entrancy in `MintingHub.challenge()`
  - Relevant external call:
    - `IERC20(position.collateral()).transferFrom(msg.sender,address(this),_collateralAmount)`
  - Event emitted after the call:
    - `ChallengeStarted(msg.sender,address(position),_collateralAmount,pos)`
- Re-entrancy in `MintingHub.clone()`
  - Relevant external call:
    - `collateral.transferFrom(msg.sender,pos,_initialCollateral)`
  - Event emitted after the call:
    - `PositionOpened(owner,address(pos),parent,address(collateral))`
- Re-entrancy in `Position.forceSale()`
  - Relevant external call:
    - `remainingCollateral = _sendCollateral(buyer,collAmount)`
      - `IERC20(collateral).transfer(target,amount)`
  - Event emitted after the call:
    - `MintingUpdate(_collateralBalance(),price,debt)`
- Re-entrancy in `Position.notifyChallengeSucceeded()`
  - Relevant external call:
    - `newBalance = _sendCollateral(_bidder,_size)`
      - `IERC20(collateral).transfer(target,amount)`
  - Event emitted after the call:
    - `MintingUpdate(newBalance,price,debt)`
- Re-entrancy in `Position.withdrawCollateral()`
  - Relevant external call:
    - `balance = _withdrawCollateral(target,amount)`
      - `IERC20(collateral).transfer(target,amount)`
  - Event emitted after the call:
    - `MintingUpdate(balance,price,principal + accruedInterest)`
- Re-entrancy in `Position.adjust()`
  - Relevant external calls:
    - `collateral.transferFrom(msg.sender,address(this),newCollateral - colbal)`
    - `_withdrawCollateral(msg.sender,colbal - newCollateral)`
      - `IERC20(collateral).transfer(target,amount)`
  - Event emitted after the calls:
    - `MintingUpdate(newCollateral,newPrice,newDebt)`

- Re-entrancy in `PositionRoller.roll()`
  - Relevant external calls:
    - `targetCollateral.transferFrom(msg.sender,address(this),collDeposit)`
    - `targetCollateral.approve(target.hub(),collDeposit)`
    - `targetCollateral.transferFrom(msg.sender,address(target),collDeposit)`
  - Event emitted after the calls:
    - `Roll(address(source),collWithdraw,repay,address(target),collDeposit,mint)`

Appendix E: Analysis of Known Solidity Compiler Bugs

The Hardhat configuration committed to the repository specifies Solidity version 0.8.26 to be used, which has no known bugs.[28] Note though that the version `pragmas` of the smart contract files specify either versions 0.8.0 or 0.8.10, which introduces the slight possibility that a lower compiler version could end up being used.

Thus, for completeness sakes, we have analyzed the union of known compiler issues that versions 0.8.10 through 0.8.28 (the most recent version) have been subject to:[29]

- `AbiEncodeCallLiteralAsFixedBytesBug`
- `AbiReencodingHeadOverflowWithStaticArrayCleanup`
- `DataLocationChangeInInternalOverride`
- `DirtyBytesArrayToStorage`
- `FullInlinerNonExpressionSplitArgumentEvaluationOrder`
- `InlineAssemblyMemorySideEffects`
- `MissingSideEffectsOnSelectorAccess`
- `NestedCalldataArrayAbiReencodingSizeValidation`
- `StorageWriteRemovalBeforeConditionalTermination`
- `VerbatimInvalidDeduplication`

None of thoses issues have been deemed relevant for the dEURO codebase.

28 https://github.com/ethereum/solidity/blob/c91c204e0ac87707547c0c1cc6e7d883c52633b6/docs/bugs_by_version.json#L1897-L1900

29 https://github.com/ethereum/solidity/blob/c91c204e0ac87707547c0c1cc6e7d883c52633b6/docs/bugs_by_version.json

Appendix F: Test Coverage

The following test coverage results have been generated by running "`npx hardhat coverage`".

File	Statements (%)	Branches (%)	Functions (%)	Lines (%)
DecentralizedEURO.sol	92.86	88.71	95.83	92.59
Equity.sol	98.94	89.66	96.67	99.06
Leadrate.sol	100	75	100	100
Savings.sol	75.68	55	72.73	81.4
StablecoinBridge.sol	100	100	100	100
MintingHub.sol	100	91.38	100	100
Position.sol	99.29	87.2	100	98.92
PositionFactory.sol	100	50	100	100
PositionRoller.sol	100	85	100	100
FrontendGateway.sol	60.61	50	70	71.93
MintingHubGateway.sol	100	50	100	100
SavingsGateway.sol	61.54	31.82	66.67	63.64
ERC3009.sol	88.24	50	80	89.47
DEPSWrapper.sol	0	0	0	0
MathUtil.sol	0	0	0	0