Report

v. 1.0

Customer

Term Structure

Smart contract and Circom Audit

Term Structure Update

8th May 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	07.05.24	A. Zveryanskaya	Initial Draft
0.2	08.05.24	A. Zveryanskaya	Minor revision
1.0	08.05.24	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

Term Structure is a decentralized bond protocol powered by zkTrue-up, a platform that enables peer-to-peer lending and borrowing with fixed interest rates.

3 Project scope

We were asked to review Term Structure update with Circom and Smart contracts code. We also checked provided:

- Circom fixes
- Solidity fixes

Reviewed Circom files:

/			
	mechanism.circom	request.circom	
const/			
	_mod.circom	fmt.circom	op_type.circom
gadgets/			
	_mod.circom		
type/			
	_mod.circom	order.circom	req.circom
zkTrueUp/			
	normal.circom	spec.circom	

Solidity files:

account/			
	AccountFacet.sol	AccountLib.sol	AccountStorage.sol
	IAccountFacet.sol		
evacuation/			
	EvacuationFacet.sol	EvacuationLib.sol	EvacuationStorage.sol
	IEvacuationFacet.sol		

extensions/sDai/		
IPot.sol	SDaiPriceFeed.sol	
extensions/wstETH/		
IWstETH.sol	wstETHPriceFeed.sol	
flashLoan/		
FlashLoanFacet.sol		
initializer/		
ZkTrueUpInit.sol		
libraries/		
Bytes.sol	Config.sol	Delegates.sol
InitialConfig.sol	Operations.sol	Signature.sol
Utils.sol		
loan/		
ILoanFacet.sol	LoanFacet.sol	LoanLib.sol
LoanStorage.sol		
rollup/		
IRollupFacet.sol	RollupFacet.sol	RollupLib.sol
RollupStorage.sol		
tsb/		
ITsbFacet.sol	TsbFacet.sol	TsbStorage.sol

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We found 8 major, and a few less important issues. or otherwise addressed in collaboration with the client.

	Info	Fixed
Major	2	6
Moderate	4	2

Fixed 8 out of 14 issues

6 Major Issues

CVF-1 FIXED

- **Category** Suboptimal
- **Source** LoanFacet.sol

Description These blocks are very similar.

Recommendation Consider extracting a function.

123
```
+{
+    AccountStorage.Layout storage asl = AccountStorage.layout();
```

134
```
+    asl.validatePermitAndIncreaseNonce(loanOwner, structHash,
   ↪ deadline, v, r, s);
 }
```

181
```
+{
+    AccountStorage.Layout storage asl = AccountStorage.layout();
```

192
```
+    asl.validatePermitAndIncreaseNonce(loanOwner, structHash,
   ↪ deadline, v, r, s);
+}
```

247
```
+{
+    AccountStorage.Layout storage asl = AccountStorage.layout();
```

257
```
+    asl.validatePermitAndIncreaseNonce(loanOwner, structHash,
   ↪ deadline, v, r, s);
+}
```

306
```
+{
+    AccountStorage.Layout storage asl = AccountStorage.layout();
```

310
```
+    asl.validatePermitAndIncreaseNonce(loanOwner, structHash,
   ↪ deadline, v, r, s);
+}
```

364
```
+{
+    AccountStorage.Layout storage asl = AccountStorage.layout();
```

368
```
+    asl.validatePermitAndIncreaseNonce(loanOwner, structHash,
   ↪ deadline, v, r, s);
+}
```

CVF-2 INFO

- **Category** Unclear behavior
- **Source** Bytes.sol

Description This reads and writes 4 bytes after the slice.

Recommendation Consider modifying the code like this: mstore(add(slice_curr, 0×1C), mload(add(array_curr, 0×1C)))

Client Comment *Limited the final length of the data returned when declaring.*

```
102 +mstore(add(slice_curr, 0x20), mload(add(array_curr, 0x20)))
```

CVF-3 INFO

- **Category** Unclear behavior
- **Source** Bytes.sol

Description This reads and writes 24 bytes after the slice.

Recommendation Consider modifying the code like this: mstore(add(slice_curr, 0×28), mload(add(array_curr, 0×28)))

Client Comment *Limited the final length of the data returned when declaring.*

```
121 +mstore(add(slice_curr, 0x40), mload(add(array_curr, 0x40)))
```

CVF-4 FIXED

- **Category** Overflow/Underflow
- **Source** SDaiPriceFeed.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Consider using the ”mulDiv” function.

```
69 +answer = (answer * _pot.chi().toInt256()) / CHI_DECIMALS;
```

CVF-5 FIXED

- **Category** Unclear behavior
- **Source** EvacuationFacet.sol

Description This will revert in case ”executedL1RequestsNum” is zero.

Recommendation Consider reverting with a meaningful error message in such a case.

```
52 +uint64 lastExecutedL1RequestId = executedL1RequestNum - 1;
```

CVF-6 FIXED

- **Category** Unclear behavior
- **Source** EvacuationFacet.sol

Description This will revert in case ”totalL1RequestNum” is zero.

Recommendation Consider reverting with a meaningful error message in such a case.

```
81 +uint64 lastL1RequestId = totalL1RequestNum - 1;
```

CVF-7 FIXED

- **Category** Documentation
- **Source** mechanism.circom

Recommendation This complicated logic should be documented.

Client Comment *Comments added and documentation updated. This code implements the 'minFee' agreement mechanism, which regulates the minimum fee for our orders. When a user signs an order, they consent to the 'minFee' amount specified as the minimum fee. Initially, upon the first match of an order, we charge an amount equivalent to the signed 'minFee' as an initial credit before proceeding with the matching process. As the order continues to match repeatedly, the transaction fee required from the user will gradually increase. However, any additional fees are only charged once the accumulated fees exceed the initial credit amount. If the fees for an order are derived from 'matchedAmt' and the 'matchedAmt' for a match is insufficient to cover the credit amount, the remaining credit will be carried over and charged in subsequent matches. A unique feature of our mechanism is that an order can assume different roles in various matches. For example, a secondary limit order might act as a taker in its first match and as a maker in later ones. In such scenarios, different 'minFee' rates may apply based on the order's role. Therefore, we include an additional rule: if the signed 'minFee' differs from the initial credit amount and 'minFee' is greater, we will charge the difference between the signed minFee and the original credit, updating the credit amount as needed. The following outlines the specific operational flow for implementing the 'minFee' mechanism:*

- *Calc 'minFee' as the maximum of the following:*
  - *'oriCreditAmt'*
  - *'signedMinFee'*
- *Calc 'newCreditAmt'*
  - *For the buy order and lend order:*
    - *'newCreditAmt := minFee'*
  - *For the sell order and borrower order:*
    - *'newCreditAmt := Min(minFee, Max(oriCreditAmt, oriCumFeeAmt) + matchedAmt)'*
- *Calc 'chargedCreditAmt' as the minimum of the following:*
  - *newCreditAmt' - 'oriCreditAmt'*
  - *'newCreditAmt' > 'oriCumFeeAmt ? newCreditAmt - oriCumFeeAmt : 0'*
- *Update the accumulated fee amount:*
  - *'newCumFeeAmt = oriCumFeeAmt + matchedFeeAmt'*
- *Calc chargedFeeAmt as the minimum of the following:*
  - *'newCumFeeAmt > newCreditAmt ? newCumFeeAmt - newCreditAmt : 0'*
  - *'matchedFeeAmt'*

- *And the 'totalChargedAmt' will be 'chargedCreditAmt + chargedFeeAmt'*
  - *If it's a borrow order or a sell order, circuit will reject when 'totalChargedAmt > matchedAmt'*

```
411 +signal newCreditAmtIfFromTarget <== Min(BitsUnsignedAmt())([
        ↪ minFeeAmt, maxOriCreditAmtOriCumFeeAmt + matched_amt1]);
    +newCreditAmt <== Mux(2)([minFeeAmt, newCreditAmtIfFromTarget],
        ↪ isFeeFromTarget);
    +ImplyEq()(enabled, 1, Or()(TagLessEqThan(BitsUnsignedAmt())([
        ↪ newCreditAmt, minFeeAmt]), TagIsEqual()([newCreditAmt,
        ↪ oriCreditAmt])));
    +ImplyEq()(enabled, 1, TagGreaterEqThan(BitsUnsignedAmt())([
        ↪ newCreditAmt, oriCreditAmt]));
    +signal sgtNewCreditAmtOriCumFeeAmt <== TagGreaterThan(
        ↪ BitsUnsignedAmt())([newCreditAmt, oriCumFeeAmt]);
    +signal chargedCreditAmt <== Min(BitsUnsignedAmt())([newCreditAmt -
        ↪ oriCreditAmt, (newCreditAmt - oriCumFeeAmt) *
        ↪ sgtNewCreditAmtOriCumFeeAmt]);
    +signal newCumFeeAmt <== oriCumFeeAmt + expectedFee;
    +signal sgtNewCumFeeAmtNewCreditAmt <== TagGreaterThan(
        ↪ BitsUnsignedAmt())([newCumFeeAmt, newCreditAmt]);
    +signal chargedFeeAmt <== Min(BitsUnsignedAmt())([expectedFee, (
        ↪ newCumFeeAmt - newCreditAmt) * sgtNewCumFeeAmtNewCreditAmt]);
420 +fee <== chargedFeeAmt + chargedCreditAmt;
```

CVF-10 FIXED

- **Category** Unclear behavior
- **Source** mechanism.circom

Description It seems weird that in some cases "newCreditAmt" should be <= "minFeeAmt".

Recommendation Consider explaining.

Client Comment *Same as the comment of CVF-7.*

```
413 +ImplyEq()(enabled, 1, Or()(TagLessEqThan(BitsUnsignedAmt())([
        ↪ newCreditAmt, minFeeAmt]), TagIsEqual()([newCreditAmt,
        ↪ oriCreditAmt])));
```

7 Moderate Issues

CVF-8 INFO

- **Category** Suboptimal
- **Source** request.circom

Description These checks make the ”channelIn” and ”channelOut” signals redundant, as these signals are always zero.

Client Comment *All templates starting with DoReq have the same input/output structure, facilitating signal chaining and Mux operations with external templates.*

1745
```
+ImplyEqArr(LenOfChannel())(enabled, channelIn, Channel_Default()())
   ↪ ;
+for(var i = 0; i < LenOfChannel(); i++)
+    channelOut[i] <== 0;
```

1924
```
+ImplyEqArr(LenOfChannel())(enabled, channelIn, Channel_Default()())
   ↪ ;
+for(var i = 0; i < LenOfChannel(); i++)
+    channelOut[i] <== 0;
```

1970
```
+ImplyEqArr(LenOfChannel())(enabled, channelIn, Channel_Default()())
   ↪ ;
+for(var i = 0; i < LenOfChannel(); i++)
+    channelOut[i] <== 0;
```

CVF-9 INFO

- **Category** Suboptimal
- **Source** request.circom

Description This check makes the ”channelOut” signal redundant, as this signals is always zero.

Client Comment *All templates starting with DoReq have the same input/output structure, facilitating signal chaining and Mux operations with external templates.*

1867
```
+for(var i = 0; i < LenOfChannel(); i++)
+    channelOut[i] <== 0;
```

CVF-11 FIXED

- **Category** Unclear behavior
- **Source** RollupLib.sol

Description This should be calculated inside the ”if” statement below. Otherwise, this will revert in case ”totalL1RequestNum” is zero.

```
191 +uint64 lastL1RequestId = totalL1RequestNum - 1;
```

CVF-12 FIXED

- **Category** Overflow/Underflow
- **Source** LoanFacet.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Consider calculating like this: rollBorrowOrder.maxBorrowAmt.mulDiv(uint256(borrowFeeRate, Config.SYSTEM_UNIT_BASE).mulDiv(maxInterestRate, Config.SYSTEM_UNIT_BASE).

```
1007 +.mulDiv(uint256(maxInterestRate) * borrowFeeRate, Config.
       ↪ SYSTEM_UNIT_BASE * Config.SYSTEM_UNIT_BASE)
```

CVF-13 INFO

- **Category** Suboptimal
- **Source** AccountFacet.sol

Description Overwriting the mask could lead to clashes between subsequent updates.

Recommendation Instead of passing the new mask consider passing two masks: one with bits to be set and another with bits to be cleared. Alternatively, consider passing the current mask, so the contract could check that the current mask it exactly what the caller expects.

```
148 +asl.delegatedActions[msg.sender][delegatee] = delegatedActions;
```

CVF-14 INFO

- **Category** Overflow/Underflow
- **Source** request.circom

Description Underflow is possible here.

Recommendation Consider implementing proper underflow protection or clearly explaining why underflow isn't actually possible.

Client Comment *The matching mechanism inherently ensures that the accumulated amounts consistently trend upwards. As a result, both 'cumAmt0' and 'cumAmt1' will consistently increase, ensuring that each new accumulated amount exceeds the previous one.*

1836
```
+var matched_amt0 = order.cumAmt0 - channel_in.args[0]/*oriCumAmt0*/
   ↪ ;
+var matched_amt1 = order.cumAmt1 - channel_in.args[1]/*oricumamt1*/
   ↪ ;
```

8 Recommendations

CVF-15 FIXED

- **Category** Readability
- **Source** TsbStorage.sol

Description Declaring a top-level constant in a file named after a library makes it harder navigating through code.

Recommendation Consider either moving the constant declaration into the library or moving it into a separate file.

```
9 +bytes32 constant REDEEM_TYPEHASH = keccak256(
```

CVF-16 INFO

- **Category** Procedural
- **Source** RollupLib.sol

Description We didn't review these files.

```
7 +import {TokenStorage, AssetConfig} from "../token/TokenStorage.sol"
    ↪ ;
```

```
10 +import {TokenLib} from "../token/TokenLib.sol";
```

CVF-17 FIXED

- **Category** Suboptimal
- **Source** RollupLib.sol

Recommendation This could be simplified as: return Operations.isRollBorrowHashedPub-DataMatched(rollBorrow, request.hashedPubData);

```
367 +if (Operations.isRollBorrowHashedPubDataMatched(rollBorrow, request
    ↪ .hashedPubData)) return true;
+return false;
```

CVF-18 FIXED

- **Category** Readability
- **Source** RollupLib.sol

Recommendation Should be ”else return”.

```
368 +return false;
```

```
382 +return false;
```

CVF-19 FIXED

- **Category** Suboptimal
- **Source** RollupLib.sol

Recommendation This could be optimized as: return Operations.isForceCancelRollBorrowHashedPubDataMatched(forceCancelRollBorrow, request.hashedPubData);

```
380 +if (Operations.isForceCancelRollBorrowHashedPubDataMatched(
    ↪ forceCancelRollBorrow, request.hashedPubData))
+     return true;
+return false;
```

CVF-20 INFO

- **Category** Procedural
- **Source** RollupFacet.sol

Description We didn’t review this file.

```
18 +import {TokenStorage, AssetConfig} from "../token/TokenStorage.sol"
    ↪ ;
```

CVF-21 FIXED

- **Category** Readability
- **Source** LoanStorage.sol

Description Declaring top-level constants in a file named after a library makes it harder navigating through code.

Recommendation Consider either moving the constants declarations into the library or moving them into a separate file.

```
57 +bytes32 constant REMOVE_COLLATERAL_TYPEHASH = keccak256(
```

```
62 +bytes32 constant REPAY_TYPEHASH = keccak256(
```

```
67 +bytes32 constant ROLL_BORROW_TYPEHASH = keccak256(
```

```
72 +bytes32 constant FORCE_CANCEL_ROLL_BORROW_TYPEHASH = keccak256(
```

```
77 +bytes32 constant ROLL_TO_AAVE_TYPEHASH = keccak256(
```

CVF-22 INFO

- **Category** Readability
- **Source** LoanStorage.sol

Description Declaring a top-level structure in a file named after a library makes it harder navigating through code.

Recommendation Consider either moving the struct declaration into the library of moving it into a separate file.

Client Comment *To maintain consistency with the declaration of all structures, keep it here.*

```
87 +struct RollBorrowOrder {
```

CVF-23 FIXED

- **Category** Bad datatype
- **Source** LoanStorage.sol

Recommendation The type of this field should be more specific.

```
93 +address tsbTokenAddr; // the tsb token address of the new term of
      ↪ the loan
```

CVF-24 FIXED

- **Category** Bad datatype
- **Source** ILoanFacet.sol

Recommendation The parameter type should be more specific.

```
15 +error InvalidTsbTokenAddr(address tsbTokenAddr);
```

CVF-25 FIXED

- **Category** Procedural
- **Source** ILoanFacet.sol

Recommendation The ”loanOwner” parameters should be indexed.

```
66 +    address loanOwner,
```

```
82 +    address loanOwner,
```

```
101 +    address loanOwner,
```

```
122 +    address loanOwner,
```

```
137 +    address loanOwner,
```

```
145 +event RollBorrowOrderForceCancelPlaced(bytes12 indexed loanId,
       ↪ address caller, address loanOwner);
```

CVF-26 INFO

- **Category** Unclear behavior
- **Source** LoanFacet.sol

Description These events are emitted even if nothing actually changed.

Client Comment *This is the admin functions and not a critical issue.*

```
449 +emit SetBorrowFeeRate(borrowFeeRate);
```

```
457 +emit SetRollOverFee(rollOverFee);
```

CVF-27 FIXED

- **Category** Suboptimal
- **Source** LoanFacet.sol

Description Handling string and bytes errors separately seems as over complication.

Recommendation Consider handling only bytes, as string error message could be extracted from a raw bytes error off-chain.

```
733 +    } catch Error(string memory err) {
+        revert BorrowFromAaveFailedLogString(supplyToken,
    ↪ collateralAmt, debtToken, debtAmt, err);
+    } catch (bytes memory err) {
+        revert BorrowFromAaveFailedLogBytes(supplyToken,
    ↪ collateralAmt, debtToken, debtAmt, err);
```

```
738 +} catch Error(string memory err) {
+    revert SupplyToAaveFailedLogString(supplyToken, collateralAmt,
    ↪ err);
740 +} catch (bytes memory err) {
+    revert SupplyToAaveFailedLogBytes(supplyToken, collateralAmt,
    ↪ err);
```

CVF-28 INFO

- **Category** Bad naming
- **Source** LoanFacet.sol

Recommendation Consider giving descriptive names to the returned values matching the names in the documentation comment.

Client Comment *No change becasue of the coding style unified*

```
756 +) internal returns (LiquidationAmt memory, IERC20) {
```

CVF-29 INFO

- **Category** Suboptimal
- **Source** LoanFacet.sol

Recommendation These two checks could be merged into one using logical ”or”.

Client Comment *No change because of the readability*

```
921 +if (rollBorrowOrder.expiredTime <= block.timestamp) revert
       ↪ InvalidExpiredTime(rollBorrowOrder.expiredTime);
    +if (rollBorrowOrder.expiredTime + Config.
       ↪ LAST_ROLL_ORDER_TIME_TO_MATURITY > oldMaturityTime)
    +    revert InvalidExpiredTime(rollBorrowOrder.expiredTime);
```

CVF-30 INFO

- **Category** Suboptimal
- **Source** LoanFacet.sol

Recommendation The formula would be simpler if a per-second interest rate would be used instead of an annual rate.

Client Comment *keep using the annual interest rate, because for uniformity, the annual interest rate is used when the user signs and our L2 system*

```
997 +.maxAnnualPercentageRate
```

CVF-31 INFO

- **Category** Procedural
- **Source** Utils.sol

Description We didn't review these files.

```
9  import {AddressStorage} from "../address/AddressStorage.sol";
10 import {AddressLib} from "../address/AddressLib.sol";
   import {TsbLib} from "../tsb/TsbLib.sol";
   import {ITsbToken} from "../interfaces/ITsbToken.sol";
```

CVF-32 INFO

- **Category** Suboptimal
- **Source** Utils.sol

Recommendation The value "10 ** decimals" should probably be passed as an argument instead of just "decimals" so the caller would have a chance to precompute it.

```
128 +return l2Amt.mulDiv(10 ** decimals, Config.SYSTEM_UNIT_BASE);
```

```
137 +return SafeCast.toUint128(l1Amt.mulDiv(Config.SYSTEM_UNIT_BASE, 10
      ↪ ** decimals));
```

CVF-33 INFO

- **Category** Procedural
- **Source** Signature.sol

Recommendation Consider specifying as "^0.8.0" unless there is something special regarding this particular version. Also relevant for: Delegate.sol, IWstETH.sol, WstETHPriceFeed.sol, IPot.sol, SDaiPriceFeed.sol, EvacuationStorage.sol, EvacuationLib.sol, IEvacuationFacet.sol, EvacuationFacet.sol.

Client Comment *Some imported package libraries > version 0.8.0, so we adopted version 0.8.17 for the consistency of the all contracts.*

```
2 +pragma solidity ^0.8.17;
```

CVF-34 FIXED

- **Category** Suboptimal
- **Source** Signature.sol

Recommendation Conversions to ”bytes” are redundant.

```
23 +bytes32 internal constant EIP712_NAME_HASH = keccak256(bytes("
      ↪ ZkTrueUp"));

25 +bytes32 internal constant EIP712_VERSION_HASH = keccak256(bytes("1"
      ↪ ));
```

CVF-35 INFO

- **Category** Readability
- **Source** Delegate.sol

Description Declaring top-level constants in a file named after a library makes it harder navigating through code.

Recommendation Consider either moving the constant declarations into the library or moving them into a separate file.

```
6  +uint256 constant DELEGATE_WITHDRAW_MASK = 1 << 255;
   +uint256 constant DELEGATE_REMOVE_COLLATERAL_MASK = 1 << 254;
   +uint256 constant DELEGATE_REPAY_MASK = 1 << 253;
   +uint256 constant DELEGATE_ROLL_TO_AAVE_MASK = 1 << 252;
10 +uint256 constant DELEGATE_ROLL_BORROW_MASK = 1 << 251;
   +uint256 constant DELEGATE_FORCE_CANCEL_ROLL_BORROW_MASK = 1 << 250;
   +uint256 constant DELEGATE_REDEEM_MASK = 1 << 249;
```

CVF-36 FIXED

- **Category** Procedural
- **Source** Delegate.sol

Recommendation Brackets are redundant.

```
19 +return (delegatedActions & actionMask) != 0;
```

CVF-37 FIXED

- **Category** Documentation
- **Source** Bytes.sol

Description There are actually only two ”mstore” opcodes.

```
100 +// mstore 3 times for 60 bytes
```

CVF-38 INFO

- **Category** Procedural
- **Source** FlashLoanFacet.sol

Description We didn’t review these files.

```
9  import {FlashLoanStorage} from "./FlashLoanStorage.sol";
10 import {ProtocolParamsStorage} from "../protocolParams/
     ↪ ProtocolParamsStorage.sol";
   import {TokenStorage} from "../token/TokenStorage.sol";
   import {FlashLoanLib} from "./FlashLoanLib.sol";
   import {IFlashLoanFacet} from "./IFlashLoanFacet.sol";
   import {IFlashLoanReceiver} from "../interfaces/IFlashLoanReceiver.
     ↪ sol";
   import {ProtocolParamsLib} from "../protocolParams/ProtocolParamsLib
     ↪ .sol";
   import {TokenLib} from "../token/TokenLib.sol";
```

CVF-39 FIXED

- **Category** Procedural
- **Source** IWstETH.sol

Recommendation This interface should be moved into a separate file named ”IStETH.sol”.

```
31 +interface IStETH is IERC20 {
```

CVF-40 FIXED

- **Category** Procedural
- **Source** WstETHPriceFeed.sol

Recommendation Brackets around multiplication are redundant.

```
69 +answer = (answer * _wstETH.stEthPerToken().toInt256()) /
      ↪ STETH_PER_WSTETH_DECIMALS;
```

CVF-41 FIXED

- **Category** Overflow/Underflow
- **Source** WstETHPriceFeed.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type while some intermediary calculation overflows.

Recommendation Consider using the ”mulDiv” function.

```
69 +answer = (answer * _wstETH.stEthPerToken().toInt256()) /
      ↪ STETH_PER_WSTETH_DECIMALS;
```

CVF-42 FIXED

- **Category** Procedural
- **Source** SDaiPriceFeed.sol

Recommendation Brackets around multiplication are redundant.

```
69 +answer = (answer * _pot.chi().toInt256()) / CHI_DECIMALS;
```

CVF-43 INFO

- **Category** Suboptimal
- **Source** EvacuationStorage.sol

Recommendation Subtracting one from a hash doesn’t have much sense, as storage layout occupies more than once slot, so the slot addressed by the original hash could also be affected.

```
9 +bytes32 internal constant STORAGE_SLOT = bytes32(uint256(keccak256(
      ↪ "zkTrueUp.contracts.storage.Evacuation")) - 1);
```

CVF-44 INFO

- **Category** Bad naming
- **Source** EvacuationLib.sol

Description The name of this error sounds like the name of an event.

Recommendation Consider renaming to emphasize what actually got wrong.

```
14 +error EvacuModeActivated();
```

CVF-45 INFO

- **Category** Bad naming
- **Source** IEvacuationFacet.sol

Description The names of these errors don't sound like errors, but rather like normal situations.

Recommendation Consider renaming to emphasize what exactly got wrong.

```
14 +error TimeStampIsNotExpired(uint256 curTimestamp, uint256
      ↪ expirationTime);

16 +error LastL1RequestIsEvacuation(uint64 totalL1RequestNum);

32 +error Evacuated(uint32 accountId, uint16 tokenId);
```

CVF-46 FIXED

- **Category** Procedural
- **Source** IEvacuationFacet.sol

Recommendation The "accountAddr" parameters should be indexed.

```
41 +    address accountAddr,

64 +event AccountDeregistered(address accountAddr, uint32 indexed
      ↪ accountId);
```

CVF-47 FIXED

- **Category** Bad naming
- **Source** IEvacuationFacet.sol

Recommendation Events are usually named via nouns, such as ”L1RequestConsumption” or ”AccountDeregistration”.

```
56 +event L1RequestConsumed(uint64 executedL1RequestNum, Operations.
      ↪ OpType opType, bytes pubData);
```

```
64 +event AccountDeregistered(address accountAddr, uint32 indexed
      ↪ accountId);
```

CVF-48 INFO

- **Category** Procedural
- **Source** EvacuationFacet.sol

Description We didn’t review these files.

```
 9 +import {AddressStorage} from "../address/AddressStorage.sol";
10 +import {TokenStorage, AssetConfig} from "../token/TokenStorage.sol"
      ↪ ;
```

```
15 +import {AddressLib} from "../address/AddressLib.sol";
   +import {TokenLib} from "../token/TokenLib.sol";
```

CVF-49 FIXED

- **Category** Readability
- **Source** AccountStorage.sol

Description Declaring a top-level constant is a file named after a library makes it harder navigating through code.

Recommendation Consider either moving the declaration into the library or moving it into a separate file.

```
 7 +bytes32 constant WITHDRAW_TYPEHASH = keccak256("Withdraw(address␣
      ↪ token,uint256␣amount,uint256␣nonce,uint256␣deadline)");
```

CVF-50 FIXED

- **Category** Unclear behavior
- **Source** AccountStorage.sol

Description Name ”isDelegated” looks like a boolean flag, while the values type for the mapping is actually ”uint256” used to store a bit mask.

Recommendation Consider rephrasing and elaborating more regarding mapping values.

```
23 +/// @notice LoanOwner => delegatee => isDelegated
```

```
26 +mapping(address => mapping(address => uint256)) delegatedActions;
```

CVF-51 FIXED

- **Category** Unclear behavior
- **Source** AccountStorage.sol

Description It is unclear what nonce is stored for an address, the last used, or the first not used or what.

Recommendation Consider explaining.

```
27 +/// @notice Mapping address to nonces for permit functions
   +mapping(address => uint256) nonces;
```

CVF-52 FIXED

- **Category** Documentation
- **Source** IAccountFacet.sol

Description In case the mask has several bits set, the semantics of the returned value is unclear.

Recommendation Consider clearly documenting.

```
97 +/// @param actionMask The mask of the action to check
   +/// @return isDelegated The isDelegated status of the account
```

CVF-53 INFO

- **Category** Procedural
- **Source** AccountLib.sol

Description We didn't review these files.

```
5 import {AddressLib} from "../address/AddressLib.sol";
```

```
8 import {AddressStorage} from "../address/AddressStorage.sol";
```

CVF-54 INFO

- **Category** Procedural
- **Source** AccountFacet.sol

Description We didn't review these files.

```
8 +import {AddressStorage} from "../address/AddressStorage.sol";
```

```
11 +import {TokenStorage, AssetConfig} from "../token/TokenStorage.sol"
   ↪ ;
```

```
14  import {TokenLib} from "../token/TokenLib.sol";
```

```
20 +import {AddressLib} from "../address/AddressLib.sol";
```

```
24  import {BabyJubJub, Point} from "../libraries/BabyJubJub.sol";
```

CVF-55 FIXED

- **Category** Bad datatype
- **Source** order.circom

Recommendation The value "4" should be a named constant.

Client Comment *To represent this logic and avoid the use of arbitrary numbers without clear explanation, use the variable 'elems' in a for loop.*

```
56 +if(i == LenOfReq() + 4){
```

CVF-56 FIXED

- **Category** Bad datatype
- **Source** order.circom

Recommendation The value ”5” should be a named constant.

Client Comment *To represent this logic and avoid the use of arbitrary numbers without clear explanation, use the variable ‘elems‘ in a for loop.*

```
70 +if(i == LenOfReq() + 5){
```

CVF-57 FIXED

- **Category** Suboptimal
- **Source** req.circom

Recommendation Instead of using a comment, just replace ”2” with a named constant.

```
16 +ImplyEq()(enabled, 1, TagLessThan(BitsTime())([currentTime *
      ↪ enabled, req_.arg[2]/*expiration time*/ * enabled]));
```

CVF-58 FIXED

- **Category** Suboptimal
- **Source** _mod.circom

Description The number 253 is prime field dependent.

Recommendation Consider using a named constant instead.

Client Comment *Improved the comment accordingly.*

```
52 +// def: if dividend is >= 2^253 or divisor = 0, then the quotient
      ↪ and remainder are both 0
```

CVF-59 FIXED

- **Category** Unclear behavior
- **Source** op_type.circom

Description Function body indentation is different for these functions.

```
7 +function OpTypeNumForceWithdraw()       { return 3; }
```

```
35 +function OpTypeNumUserCancelRollBorrow()  { return 30; }
+function OpTypeNumAdminCancelRollBorrow() { return 31; }
+function OpTypeNumForceCancelRollBorrow() { return 32; }
```

CVF-60 FIXED

- **Category** Bad datatype
- **Source** mechanism.circom

Recommendation The index ”8” should be a named constant.

```
375 +signal is2ndBuy <== And()(isSecondary, Not()(Bool()(req.arg[8])));
+signal is2ndSell <== And()(isSecondary, Bool()(req.arg[8]));
```

CVF-61 FIXED

- **Category** Suboptimal
- **Source** mechanism.circom

Recommendation It would be more efficient to use a version of ”And” with three arguments.

```
515 +isMatched <== And()(TagIsEqual()([lend_req.arg[1]/* maturity time
   ↪ */, borrow_req.arg[1]/* maturity time */]), And()(TagIsEqual()
   ↪ ([lend_req.tokenId, borrowingTokenId]), TagGreaterEqThan(
   ↪ BitsRatio())([borrow_req.arg[3]/* PIR */, lend_req.arg[3]/*
   ↪ PIR */])));
```

CVF-62 FIXED

- **Category** Bad datatype
- **Source** request.circom

Recommendation The value ”86400” should be a named constant.

```
675 +ImplyEq()(isAuc, 1, TagLessEqThan(BitsTime() + 1)([req.arg[2] +
       ↪ 86400, tSBToken.maturity]));
```

```
1555 +ImplyEq()(enabled, 1, TagGreaterThan(BitsTime() + 1)([p_req.
       ↪ matchedTime[0] + 86400 * upper_lim_of_days, tSBToken.maturity
       ↪ ]));
```

CVF-63 FIXED

- **Category** Bad datatype
- **Source** request.circom

Recommendation The value ”366” should be a named constant.

```
684 +ImplyEq()(isAuc, 1, TagGreaterEqThan(BitsRatio() + BitsTime() + 1)
       ↪ ([req.arg[3]/*PIR*/ * daysFromMatchedIfEnabled + 366 * one,
       ↪ daysFromMatchedIfEnabled * one + req.arg[3]/*PIR*/]));
```

CVF-64 FIXED

- **Category** Bad datatype
- **Source** request.circom

Recommendation The value ”365” should be a named constant.

```
691 +ImplyEq()(is2nd, 1, TagGreaterThan(BitsUnsignedAmt() + BitsTime() +
       ↪  1)([MQ * daysFromMatchedIfEnabled + 365 * BQ, BQ *
       ↪ daysFromMatchedIfEnabled]));
```

CVF-65 FIXED

- **Category** Suboptimal
- **Source** request.circom

Description The value ”isAuction * 1 + isRoll * 2” is calculated twice.

Recommendation Consider calculating once and reusing.

974
```
+new_order0.arr <== Multiplexer(LenOfOrderLeaf(), 3)([newTaker,
   ↪ newBorrow, newBorrowIfRoll], isAuction * 1 + isRoll * 2);
```

977
```
+new_order1.arr <== Multiplexer(LenOfOrderLeaf(), 3)([newMaker,
   ↪ newLend, newLendIfRoll], isAuction * 1 + isRoll * 2);
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting