#HashEx
BLOCKCHAIN SECURITY

Fringe Finance

smart contracts final audit report

February 2024


hashex.org


contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Fringe Finance team to perform an audit of their smart contract. The audit was conducted between 21-08-2023 and 13-02-2024.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at https://github.com/fringe-finance/primary-smart-contracts GitHub repository and was audited after the commit 86ece1a.

Update: the Fringe Finance team has responded to this report. The updated code is located in @fringe-finance/primary-smart-contracts GitHub repository after the 1c727f6 commit. Updated contracts are deployed to the multiple chains with upgradable proxies. Deployed proxy addresses are available in the documentation section of the Fringe's website. Proxies' implementations:

	Ethereum	Arbitrum	Polygon	Optimism	zk-Sync
PrimaryLendingPlatformProxyAdmin	0xe386b53Fa76C0624A27AB2a45Cc6848009f927D8	0x43db56f3bf2D8491e5abe406F8002F0Cf1A2579c	0x43db56f3bf2D8491e5abe406F8002F0Cf1A2579c	0x43db56f3bf2D8491e5abe406F8002F0Cf1A2579c	0xbBDa9272ef75Ff13462AdD78917E747e1BD32D78

PythPriceProvider	0xC20E2debccfBF359E3EB9FF5022D11742Ad63C07	0xFdab306083DA7903F394Bd1d4B105ccc792fc10e	0xDACe95F60b3D63859f9996D0dEe2014906f091e1	0x18B228417d74011e204E161857D2BB7f1B44Ec54	0x6723139cd9b99E0fb2Dd8a9A91A08AF91d4F060A
ChainlinkPriceProvider	0xFf873D2a55Be5b71E6F3FcA5259c914baD6491A3	0xC6Cda4538cec59924B7e3CC1C94f460167F8b595	0xC6Cda4538cec59924B7e3CC1C94f460167F8b595	0xC6Cda4538cec59924B7e3CC1C94f460167F8b595	
PriceProviderAggregator	0x25DF082b312e92b57f31678B171dE18dbF026011	0x1337272eC57e70fB35dBeD14acAf733930299B0e	0xfe3e03eea8C3e4f885e8F55B040FACB9ABAD7f67	0x5a9F66cdbea7c3E33EaBEF562F82b8465101D9C4	0xD04597bBa89940998C703a751e7Cb4A731350397
wstETHPriceProvider	0xA473fe03308d822725d44ED6DA555A73a3f3F65C				
Bondtroller	0x32C1efeb27BeEfE42cf6763ADd8c65DAbfC78CD3	0x805A0d736B24e276Ca2AbFfe3B4941c2413BbF2f	0x88dd5e2d05650A898821CC292561502ADEF1b255	0x18126FEf4cd8700DB1Ae6f8fD1a8c9F5159e06f2	0x69B6041eE0be4CaC8621B92ADD4268191516C678A
BLendingToken	0x3eBa0CF300d02E8A51a22093b693Fbc3Ea299Bd2	0x68cB49540d2BcA2a11f36dE0C8a4524e305e6E2E	0x139c5c8Dd5a7378076F5FE9cC4fAf67B372B2712	0x505C873EE726F5008882c2B28C9a23dF133A30C8	0xA0092C38C0E21c48C900A5193fc7aeD19C428c18
PrimaryLendingPlatformV2 Zksync	0x2af24C7a11Ad1d03E26083462F17548aB99088D2	0x08e51a50bA708e96629767c637176A47AF92f9A1	0xa5fcDbCd72fE6755779Ca1897Ef4EC961a249Ee3	0x739d83252ff5AF7f87427D991d605a5854480d35	0x119EEbe2A461f04F9f2c91453D67F96D1b7C43ef

PrimaryLendingPlatformAtomicRepaymentZksync	0xDCBFA4b0AD46839af4A7D7ADA40F798B0BC5F2dA	0x347b2eA87040624c5BB8B505688d94CF82ebc16E	0x1e3E58288AD837D13D488DA2871033581f1BB673	0xb28B184d25A1f409FF4789F9EF0850E28431CD5e	0x230E76cdeAa9Dd553B19DDB13b7DED3E0563Ec4A
PrimaryLendingPlatformLiquidationZksync	0x25ff71863fc45C0c235563041Eea10253463e601	0x188ef86177df623b0580ED22316208dD8DB90878	0x3D3282d594DdFFA60f155CB2e2D37234CD829CB6	0x77C33773F10788a5cd0a9De02ceAcc79975dA0b3	0x2d6D68Ab407e5b01AEd6Dfc5a961E1A7EE08e63D
PrimaryLendingPlatformModerator	0xA7d8EA7f0a958252E832Faa3AdD89E6e25D29682	0x1ABEFCBa8352e3d845647B7f73e561BD73A0f3Dc	0x1C680f4DCee61664b2e8664095e36128B6F084b7	0x853376f0b1572E7D619A41AAAbE53D0c944c16f5	0x1339f081a025Ea6C30b6C264f3306d2f22fEa0f1
PrimaryLendingPlatformWrappedTokenGatewayZksync	0xa79538dbA3b91d6872b5440a7b4703f559402D82	0xcBc58208A65800AbF8A03D7f177Ff9680f3A6E6B	0x3aB98b39A07906AD05e92f1924130e73bab228fF	0x3aB98b39A07906AD05e92f1924130e73bab228fF	0x24384d28Bd39384C5b9961a577c618aEbe131555
PrimaryLendingPlatformLeverageZksync	0x1c54308C4B43CE6b5475193a6B2bea0324e1D37b	0x3f70C28d06Ca8e15AdCe916E0cD1D930d3409B33	0xb9CF4AC02780B993Ad6095142241b84e3DaC075e	0xdeA5b494759C46AC3F9ecA6C899054771F42b3A6	0xcB6f5a4cD75bE3b149a989f32fB43E9bA2aC74De

JumpRateModelV3	0x03a7984F9985391ec7B75fb9bB8CEfC97EAE9Cc3	0xff2c24F870a39b91c55c9d271da25EE7A6a7E9D7	0xdfe98145Aa50fA6f422cB46A9993F0f867ACC142	0x286475366f736fcEeB0480d7233ef169AE614Fe4	0x8d39aBc1Ae6FDCb4a46Ba7B76177DEde9B8d5351

2.1 Summary

Project name	Fringe Finance
URL	https://fringe.fi
Platform	Ethereum, zkSync, Polygon Network, Arbitrum Network, Optimism
Language	Solidity
Centralization level	High
Centralization risk	High

2.2 Contracts

Name	Address
PrimaryLendingPlatformV2Core	
BLendingToken	0x3eBa0CF300d02E8A51a22093b693Fbc3Ea299Bd2
BToken	
PrimaryLendingPlatformLiquidationCore	
PrimaryLendingPlatformAtomicRepaymentCore	
PrimaryLendingPlatformModerator	

PrimaryLendingPlatformWrappedTokenGatewayCore	
PrimaryLendingPlatformLeverageCore	
PrimaryLendingPlatformV2	
PrimaryLendingPlatformLiquidation	
PrimaryLendingPlatformAtomicRepayment	
PrimaryLendingPlatformWrappedTokenGateway	
PrimaryLendingPlatformLeverage	
PrimaryLendingPlatformV2Zksync	0x2af24C7a11Ad1d03E26083462F17548aB99088D2
PrimaryLendingPlatformLiquidationZksync	0x25ff71863fc45C0c235563041Eea10253463e601
PrimaryLendingPlatformAtomicRepaymentZksync	0xDCBFA4b0AD46839af4A7D7ADA40F798B0BC5F2dA
PrimaryLendingPlatformWrappedTokenGatewayZksync	0xa79538dbA3b91d6872b5440a7b4703f559402D82
PrimaryLendingPlatformLeverageZksync	0x1c54308C4B43CE6b5475193a6B2bea0324e1D37b
PriceProviderAggregatorPyth	0x25DF082b312e92b57f31678B171dE18dbF026011
ChainlinkPriceProvider	0xFf873D2a55Be5b71E6F3FcA5259c914baD6491A3
ChainlinkPriceProviderL2	0xC6Cda4538cec59924B7e3CC1C94f460167F8b595
PythPriceProvider	0xC20E2debccfBF359E3EB9FF5022D11742Ad63C07
JumpRateModelV3	0x03a7984F9985391ec7B75fb9bB8CEfC97EAE9Cc3

InterestRateModel	
LPPriceProvider	
wstETHPriceProvider	0xA473fe03308d822725d44ED6DA555A73a3f3F65C

3. Project centralization risks

The reviewed contract is highly dependent on the owner's account. The contracts are designed to be upgradeable meaning that accounts with privileged access can change implementations of the contracts. Users using the project have to trust the owner and that the owner's account is properly secured.

Fringe team response

To mitigate governance risks, all changes to Fringe's contracts are subject to a time-delay, so that users are given an opportunity to react before they come into effect. Though Fringe's time delay is currently set at zero, as the current parameters become time-tested and Fringe's TVL increases, Fringe's time delay will be progressively revised to one week and potentially longer.

4. Found issues

Critical	2 (9%)
Low	11 (48%)
Info	10 (43%)

C01. PrimaryLendingPlatformV2Core

ID	Severity	Title	Status
C01I6e	Low	Duplicated modifier	Resolved
C01I72	Low	Gas optimizations	Resolved
C01I73	Info	Stored interest may be outdated	Acknowledged
C01I74	Info	Lack of events	Acknowledged

C02. BLendingToken

ID	Severity	Title	Status
C02I80	Low	Unused code	Resolved
C02I69	Info	Typos	Resolved

C03. BToken

ID	Severity	Title	Status
C03I75	Info	Disabled safety check	Resolved

C05. PrimaryLendingPlatformAtomicRepaymentCore

ID	Severity	Title	Status
C05I6a	Low	Incorrect usage of OpenZeppelin's safe approve	Resolved

C08. PrimaryLendingPlatformLeverageCore

ID	Severity	Title	Status
C08I7e	Low	Gas optimizations	Resolved
C08I6b	Low	Incorrect usage of OpenZeppelin's safe approve	Resolved

C09. PrimaryLendingPlatformV2

ID	Severity	Title	Status
C09I70	Critical	Lack of access modifier for withdrawFromRelatedContracts function	Resolved

C0e. PrimaryLendingPlatformV2Zksync

ID	Severity	Title	Status
C0eI6f	Critical	Lack of access modifier for withdrawFromRelatedContracts function	Resolved

C10. PrimaryLendingPlatformAtomicRepaymentZksync

ID	Severity	Title	Status
C10I6c	Low	Incorrect usage of OpenZeppelin's safe approve	Resolved

C12. PrimaryLendingPlatformLeverageZksync

ID	Severity	Title	Status
C12I6d	Low	Incorrect usage of OpenZeppelin's safe approve	Resolved

C14. ChainlinkPriceProvider

ID	Severity	Title	Status
C14I81	Low	Typo	Resolved

C17. JumpRateModelV3

ID	Severity	Title	Status
C17I77	Low	Gas optimizations	Acknowledged
C17I78	Info	Typos	Resolved

C18. InterestRateModel

ID	Severity	Title	Status
C18I76	Info	Irrelevant NatSpec description	Resolved

C19. LPPriceProvider

ID	Severity	Title	Status
C19I82	Info	Typos	Resolved
C19I79	Info	Tokens with transfer hooks aren't supported	Acknowledged
C19I7a	Info	Feeds decimals aren't checked	Resolved

C1a. wstETHPriceProvider

ID	Severity	Title	Status
C1aI7b	Low	Gas optimizations	Resolved
C1aI7c	Info	Chainlink answer is not checked	Resolved

5. Contracts

C01. PrimaryLendingPlatformV2Core

Overview

The primary contract serves as the core engine for Fringe Finance, facilitating the deposit and withdrawal of project tokens, as well as managing the lending, borrowing, and repayment of tokens. The contract is designed as an abstract contract, it is intended to be extended by chain-specific implementations.

Issues

C01I6e Duplicated modifier

Low Resolved

The function `calcDepositPosition()` has duplicated modifier `onlyRelatedContracts`.

```
    function calcDepositPosition(
        address projectToken,
        uint256 projectTokenAmount,
        address user
    ) external isProjectTokenListed(projectToken) onlyRelatedContracts
onlyRelatedContracts nonReentrant {
        _calcDepositPosition(projectToken, projectTokenAmount, user);
    }
```

Recommendation

Remove the duplicated modifier.

C01I72 Gas optimizations

Low Resolved

1. Unnecessary action in the `removeProjectToken()` and `removeLendingToken()` functions: delete of token info array items before re-writing them.

2. Double call of `getTokenEvaluation()` in the `getCollateralAvailableToWithdraw()` function.
3. Multiple reads from storage in the `_supply()`, `_redeem()`, `_redeemUnderlying()` functions: `lendingTokenInfo[lendingToken].bLendingToken` variable.
4. Multiple reads from storage in the `_repay()`, `updateInterestInBorrowPositions()`, `getPosition()` functions: `projectTokens.length` variable.
5. Multiple reads from storage in the `_repayFully()` function: `borrowPositionInfo.loanBody` variable.

C01I73 Stored interest may be outdated

● Info ⊘ Acknowledged

The `getPosition()` function reads `bLendingToken.getEstimatedBorrowBalanceStored()`, which is a view function that doesn't accrue interest.

Any contracts should update interest prior reading `getPosition()` or `healthFactor()`: `PrimaryLendingPlatformLiquidationCore.getCurrentHealthFactor()`, `PrimaryLendingPlatformLiquidationCore.getHf()`,

`PrimaryLendingPlatformAtomicRepaymentCore.getTotalOutstanding()`, `PrimaryLendingPlatformLeverageCore._checkIsValidPosition()`.

Team response

`getEstimatedBorrowBalanceStored` is not a function to accrue interest. It only estimates the borrow balance of an account based on the current borrow index (this is one of the steps to calculate accrued interest), calculating accrued interest is in the next part of the function.

`getPosition` is the estimated function to return the details of a user's borrow position for a specific project token and lending token right at the time of call.

Recommendation

We can't ensure correctness of the estimation in all possible cases, that's why we suggest that any function, requiring a solid position data, should accrue interest explicitly via `bToken` contract.

C01I74 Lack of events

● Info ✓ Acknowledged

The functions `setPriceOracle()`, `setPrimaryLendingPlatformLeverage()`, `removeProjectToken()`, `removeLendingToken()`, `setBorrowLimitPerCollateralAsset()`, `setBorrowLimitPerLendingAsset()`, `setProjectTokenInfo()`, `setPausedProjectToken()`, `setLendingTokenInfo()`, `setPausedLendingToken()` don't emit events, complicating off-chain tracking of changes.

Team response

• With `setProjectTokenInfo()`, `setLendingTokenInfo()`: it's not necessary to emit an event.

• For the remaining functions: `setPriceOracle()`, `setPrimaryLendingPlatformLeverage()`, `removeProjectToken()`, `removeLendingToken()`, `setBorrowLimitPerCollateralAsset()`, `setBorrowLimitPerLendingAsset()`, `setPausedProjectToken()`, `setPausedLendingToken()` events are emitted through the Moderator contract (another reason is to avoid the risk of stacking too deep for PrimaryLendingPlatformV2Core).

C02. BLendingToken

Overview

A High-level contract performing borrows and redemption of tokens. The functions performing borrows and redemptions can only be called by the PrimaryLendingContract. The contract uses access control for admin and moderator roles. Accounts with the admin role can update the address of the PrimaryLendingContract.

Issues

C02I80 Unused code

Low Resolved

The moderator role and the `onlyModerator()` modified are not used within the contract and can be deleted.

```
modifier onlyModerator() {
    require(hasRole(MODERATOR_ROLE, msg.sender), "msg.sender not moderator");
    _;
}
```

The only function using the onlyModerator() modifier is commented out.

```
/********************** MODERATOR FUNCTIONS ********************** */

// function setReserveFactor(uint256 reserveFactorMantissa) public onlyModerator{
//     _setReserveFactorFresh(reserveFactorMantissa);
// }
```

Recommendation

Remove the unused code.

C02I69 Typos

Info Resolved

There is a typo in the function naming: `grandModerator()` should be `grantModerator()`.

C03. BToken

Overview

Fork of Compound V2's CToken. An abstract contract implementing logic for minting bTokens for underlying assets and calculation of interest.

Issues

C03I75 Disabled safety check

Info Resolved

The safety check for the `borrowRateMantissa` is disabled in the `accrueInterest()` function.

```
// require(borrowRateMantissa <= borrowRateMaxMantissa, "borrow rate is absurdly high");
```

C04. PrimaryLendingPlatformLiquidationCore

Overview

An abstract contract that implements the logic for partial liquidations of borrowers' positions. The partial liquidation closes only part of a position needed to bring the position to a healthy state. The contract is intended to be extended by chain-specific implementations.The contract uses an access control mechanism for admin and moderator roles. Accounts with the moderator role can set minimum partial liquidation amount, maximum partial liquidation amount, liquidator reward calculator factor, and update the PrimaryLedningPlatform contract address. The admin role is not used.

C05. PrimaryLendingPlatformAtomicRepaymentCore

Overview

An abstract contract implementing logic for atomic loan repayments. Swaps collateral tokens on a DEX for borrowed tokens to repay a loan. The contract is intended to be extended by chain-specific implementations.

The contract uses an access control mechanism for admin and moderator roles. Accounts with the moderator role can update the PrimaryLedningPlatform contract address. The admin role is not used.

Issues

C05I6a Incorrect usage of OpenZeppelin's safe approve

Low Resolved

The function `_approveTokenTransfer()` checks if there is enough approval for `tokenTransferProxy` address and if the approved amount is not enough, it approves maximum amount. In case there is some amount less than `tokenAmount` already approved for the `tokenTransferProxy`, this function will fail because OpenZeppelin's `safeApprove` fails if there is any approved amount.

```
    function _approveTokenTransfer(address token, uint256 tokenAmount) internal virtual {
        address tokenTransferProxy =
IParaSwapAugustus(exchangeAggregator).getTokenTransferProxy();
        if (ERC20Upgradeable(token).allowance(address(this), tokenTransferProxy) <=
tokenAmount) {
            ERC20Upgradeable(token).safeApprove(tokenTransferProxy, type(uint256).max);
        }
    }
```

It must be noted that in the current implementation only maximum value of `uint256` could be set for approval, but we strongly recommend updating to code to prevent any issues with subsequent code updates.

Also, the NatSpec documentation of the function states that is approves the specified amount of tokens but not the maximum value of `uint256`.

On the other hand, the `_repayAtomic()` function doesn't use SafeERC20 at all.

Recommendation

Increase appove amount instead of using `safeApprove()` for maximum amount.Update the documentation of the function to conform to the actual behavior.

C06. PrimaryLendingPlatformModerator

Overview

The contract performs administrative tasks for the Fringe Finance lending platform: adding and removing project tokens, pausing protocol, updating protocol parameters.

C07. PrimaryLendingPlatformWrappedTokenGatewayCore

Overview

A helper contract that automatically wraps ETH to wETH to use on the platform and unwraps on the way out.

The contract uses an access control mechanism for admin and moderator roles. Accounts with the moderator role can update the PrimaryLedningPlatform, PrimaryLendingPlatformLiquidation, PrimaryLendingPlatformLeverage contract addresses. The admin role is not used.

C08. PrimaryLendingPlatformLeverageCore

Overview

An abstract contract implementing logic for leveraged trading. The main function is leveragedBorrow which performs the borrow of assets, then swaps the borrowed assets on exchange aggregator to the project token to use them as collateral. If needed it transfers from a user previously approved tokens to conform to the requirements of the health of the position.

The contract is intended to be extended by chain-specific implementations.

Issues

C08I7e Gas optimizations

Low Resolved

1. Unnecessary code in the `isValidCollateralization()` function: `isValid = ratioNumerator > ratioDenominator ? true : false`.

C08I6b Incorrect usage of OpenZeppelin's safe approve

Low Resolved

The function `_approveTokenTransfer()` checks if there is enough approval for tokenTransferProxy address and if the approved amount is not enough, it approves maximum amount. In case there is some amount less than tokenAmount already approved for the tokenTransferProxy, this function will fail because OpenZeppelin's safeApprove fails if there is any approved amount.

```
    function _approveTokenTransfer(address token, uint256 tokenAmount) internal virtual {
        address tokenTransferProxy =
IParaSwapAugustus(exchangeAggregator).getTokenTransferProxy();
        if (ERC20Upgradeable(token).allowance(address(this), tokenTransferProxy) <=
tokenAmount) {
            ERC20Upgradeable(token).safeApprove(tokenTransferProxy, type(uint256).max);
        }
    }
```

It must be noted that in the current implementation only maximum value of `uint256` could be

set for approval, but we strongly recommend updating to code to prevent any issues with subsequent code updates.

Also, the NatSpec documentation of the function states that is approves the specified amount of tokens but not the maximum value of `uint256`.

Recommendation

Increase appove amount instead of using `safeApprove()` for maximum amount.Update the documentation of the function to conform to the actual behavior.

C09. PrimaryLendingPlatformV2

Overview

Implementation of the PrimaryLendingPlatformCoreV2 for the Ethereum network. Inherits from the PrimaryLendingPlatformCoreV2Core and calls its internal functions `_borrow()` and `_withraw()` with additional checks such as checking access control and checking that project and lending tokens are listed on the platform.

Issues

C09I70 Lack of access modifier for withdrawFromRelatedContracts function

Critical Resolved

The function `withdrawFromRelatedContracts()` is aimed to allow a related contract to initiate a withdrawal of a given amount of a project token from a user's deposit position.

```
    function withdrawFromRelatedContracts(
        address projectToken,
        uint256 projectTokenAmount,
        address user,
        address beneficiary,
        bytes32[] memory priceIds,
```

```
        bytes[] calldata updateData
    ) external payable isProjectTokenListed(projectToken) nonReentrant returns (uint256) {
        priceOracle.updatePrices{value: msg.value}(priceIds, updateData);
        return _withdraw(projectToken, projectTokenAmount, user, beneficiary);
    }
```

The function lacks any access modifiers and could be called by anyone.

Recommendation

Add `onlyRelatedContracts` modifier to the function declaration.

C0a. PrimaryLendingPlatformLiquidation

Overview

Implementation of the PrimaryLendingPlatformLiquidationCore contract for the Ethereum network. Inherits from PrimaryLendingPlatformLiquidationCore contract and calls its internal `_liquidate()` function with access control check and checks that project and lending tokens are listed.

C0b. PrimaryLendingPlatformAtomicRepayment

Overview

Implementation of the PrimaryLendingPlatformAtomicRepaymentCore for the Ethereum network. Has a privileged function accessible by accounts with the moderator role to set an exchange aggregator contract. Provides a public interface for atomic repayment.

C0c. PrimaryLendingPlatformWrappedTokenGateway

Overview

Implementation of the PrimaryLendingPlatformWrappedTokenGatewayCore contract for the Ethereum network.

C0d. PrimaryLendingPlatformLeverage

Overview

Implementation of the PrimaryLendingPlatformLeverageCore contract for the Ethereum network.

C0e. PrimaryLendingPlatformV2Zksync

Overview

Implementation of the PrimaryLendingPlatformLeverageCore contract for the zkSync network. Updates oracle prices before calling inherited functions.

Issues

C0el6f Lack of access modifier for withdrawFromRelatedContracts function

Critical Resolved

The function `withdrawFromRelatedContracts()` is aimed to allow a related contract to initiate a withdrawal of a given amount of a project token from a user's deposit position.

```
    function withdrawFromRelatedContracts(
        address projectToken,
        uint256 projectTokenAmount,
        address user,
```

```
        address beneficiary,
        bytes32[] memory priceIds,
        bytes[] calldata updateData
    ) external payable isProjectTokenListed(projectToken) nonReentrant returns (uint256) {
        priceOracle.updatePrices{value: msg.value}(priceIds, updateData);
        return _withdraw(projectToken, projectTokenAmount, user, beneficiary);
    }
```

The function lacks any access modifiers and could be called by anyone.

Recommendation

Add `onlyRelatedContracts` modifier to the function declaration.

C0f. PrimaryLendingPlatformLiquidationZksync

Overview

Implementation of the PrimaryLendingPlatformLiquidationCore contract for the zkSync network. Updates oracle prices before calling inherited functions.

C10. PrimaryLendingPlatformAtomicRepaymentZksync

Overview

Implementation of the PrimaryLendingPlatformAtomicRepaymentCore contract for the zkSync network. Updates oracle prices before calling inherited functions.

Issues

C10I6c Incorrect usage of OpenZeppelin's safe approve

Low Resolved

The function `_approveTokenTransfer()` checks if there is enough approval for exchangeAggregator address and if the approved amount is not enough, it approves maximum amount. In case there is some amount less than tokenAmount already approved for

the exchangeAggregator, this function will fail because OpenZeppelin's safeApprove fails if there is any approved amount.

```
    function _approveTokenTransfer(address token, uint256 tokenAmount) internal override {
        if (ERC20Upgradeable(token).allowance(address(this), exchangeAggregator) <=
tokenAmount) {
            ERC20Upgradeable(token).safeApprove(exchangeAggregator, type(uint256).max); //
@audit can fail in approve for 1 token
        }
    }
```

It must be noted that in the current implementation only maximum value of `uint256` could be set for approval, but we strongly recommend updating to code to prevent any issues with subsequent code updates.

Also, the NatSpec documentation of the function states that is approves the specified amount of tokens but not the maximum value of `uint256`.

Recommendation

Increase appove amount instead of using `safeApprove()` for maximum amount.Update the documentation of the function to conform to the actual behavior.

C11. PrimaryLendingPlatformWrappedTokenGatewayZksync

Overview

Implementation of the PrimaryLendingPlatformWrappedTokenGatewayCore contract for the zkSync network.

C12. PrimaryLendingPlatformLeverageZksync

Overview

Implementation of the PrimaryLendingPlatformLeverageCore contract for the zkSync network. Updates oracle prices before calling inherited functions.

Issues

C12I6d Incorrect usage of OpenZeppelin's safe approve

Low Resolved

The function `_approveTokenTransfer()` checks if there is enough approval for exchangeAggregator address and if the approved amount is not enough, it approves maximum amount. In case there is some amount less than tokenAmount already approved for the exchangeAggregator, this function will fail because OpenZeppelin's safeApprove fails if there is any approved amount.

```
    function _approveTokenTransfer(address token, uint256 tokenAmount) internal override {
        if (ERC20Upgradeable(token).allowance(address(this), exchangeAggregator) <=
tokenAmount) {
            ERC20Upgradeable(token).safeApprove(exchangeAggregator, type(uint256).max); //
@audit can fail in approve for 1 token
        }
    }
```

It must be noted that in the current implementation only maximum value of `uint256` could be set for approval, but we strongly recommend updating to code to prevent any issues with subsequent code updates.

Also, the NatSpec documentation of the function states that is approves the specified amount of tokens but not the maximum value of `uint256`.

Recommendation

Increase appove amount instead of using `safeApprove()` for maximum amount.Update the documentation of the function to conform to the actual behavior.

C13. PriceProviderAggregatorPyth

Overview

A price oracle for Pyth network. Inherits from PriceProviderAggregator and has functions to update Pyth price provide address and functions for updating prices and getting expired price feeds.

C14. ChainlinkPriceProvider

Overview

A price provider for Chainlink service.

Issues

C14I81 Typo

Low Resolved

There is a typo in the function naming: `grandModerator()` should be `grantModerator()`.

C15. ChainlinkPriceProviderL2

Overview

A price provicer for Chainlink service.

C16. PythPriceProvider

Overview

A price provider for Pyth service.

C17. JumpRateModelV3

Overview

Borrow interest rate model.

Issues

C17I77 Gas optimizations

Low Acknowledged

1. Unnecessary reads from storage in the `grandModerator()` and `revokeModerator()` functions: msg.sender's role is checked twice.

C17I78 Typos

Info Resolved

Typos reduce the code's readability. Typos in 'grandModerator', 'Suport', 'suply'.

C18. InterestRateModel

Overview

An abstract contract for interest model contracts. Forked from Compound V2.

Issues

C18I76 Irrelevant NatSpec description

Info Resolved

The description of the `storeBorrowRate()` function is copied from the `getBorrowRate()` function.

The `getBorrowRate()` and `getSupplyRate()` functions both have incomplete descriptions: `blendingToken` parameter is missing.

C19. LPPriceProvider

Overview

A price provider for Uniswap-like pair tokens.

Issues

C19I82 Typos

Info Resolved

There is a typo in the function naming: `grandModerator()` should be `grantModerator()`.

C19I79 Tokens with transfer hooks aren't supported

Info Acknowledged

The `getUSDPx()` function reads pair's reserves, which may be inaccurate if any of pair's underlying tokens is a token with transfer hooks.

C19l7a Feeds decimals aren't checked

Info Resolved

The `calcUSDPx112()` function implicitly calculates `PriceProvider.getPrice()` for both pair's `token0` and `token1` to have same decimals `usdDecimals` value.

C1a. wstETHPriceProvider

Overview

A price provider for the wrapped stETH.

Issues

C1al7b Gas optimizations

Low Resolved

1. Duplicated functions `isListed()` and `isActive()`.

C1al7c Chainlink answer is not checked

Info Resolved

Chainlink freshness is not checked in the `getPriceSTETH()` function.

Chainlink answer's decimals are not used in calculations.

6. Conclusion

2 critical, 11 low severity issues were found during the audit. 2 critical, 10 low issues were resolved in the update. The reviewed contracts are highly dependent on the owner's account. See the centralization risks chapter.

This audit includes recommendations on code improvement and the prevention of potential attacks.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY