MUNDUS
SECURITY

ZeroLend deployment check

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.
The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Reference information

Name	ZeroLend
Language	Solidity
Chain	zkSync Era
Website	https://zerolend.xyz/
Documentation	https://docs.zerolend.xyz/
Reference repositories	https://github.com/zerolend/core-contracts https://github.com/zerolend/periphery-contracts https://github.com/zerolend/pyth-oracles

Scope of work

Core contracts - 1

contract	address	verified
AaveOracle	0x785765De3E9ac3D8eEb42B4724A7FEA8990142B8	True
ACLManager	0x9A60cce3da06d246b492931d2943A8F574e67389	True
AToken	0xe8178fF950Ea1B69a51cE961C542a4CC6Cb6e38E	True
BorrowLogic	0x07c9C19a4823f7F89eE63cb0d89AEF55F4D61f71	False
BridgeLogic	0xeb3A0D513F497cE6E61278B628bb56470f7b357f	False
ConfiguratorLogic	0xC504e8FB2f8D76fef6Ce251A3760a507837E38f5	False
DelegationAwareAToken	0x102699803F4A2b02046C38C672401759af633510	True
EModeLogic	0x3733D1faE7965b573C018c4e65Bc4a1389cD4393	False
FlashLoanLogic	0x24Bb7d14Aad51Cbf4f187a27EF72C77231E9e5f0	False
LiquidationLogic	0xC2ec0e44a0F8262757f569942bE474e70411a85c	False
PoolAddressesProvider Registry	0x78B93fBb35C97b32C7381C81Fa3A620b3fB7787B	False
PoolAddressesProvider	0x4f285Ea117eF0067B59853D6d16a5dE8088bA259	True
PoolConfigurator-Implementation *Proxy*	0x3d8Cb6c7b4679c56EdF89050f66751e6c5D24502 0x9C3058F7bfCA6139ac3013999F57D7aa6a3AB1Ed	True
PoolDataProvider	0xB73550bC1393207960A385fC8b34790e5133175E	True
Pool-Implementation *Proxy*	0xEA56De428cB2eFdec7B11a4bB2985A0CeE3Dfd6f 0x4d9429246EA989C9CeE203B43F6d1C7D83e3B8F8	True
PoolLogic	0x969a8A5a56B82914775F5c704348594327e28EF5	False

Core contracts - 2

contract	address	verified
ReservesSetupHelper	0xe00d794744e763BeC67BdEdF6e852D4e0d958DFb	True
ReserveStrategy-rateStrategyStableOne	0x70cA80C5dE9fC8f080a494453dF1aA9180073031	True
ReserveStrategy-rateStrategyStableTwo	0xcaA502e289bFb924732f44f5E70bd08fc052aab8	True
ReserveStrategy-rateStrategyVolatileOne	0xEdAc06D73DbdD3460B5728E4bBE9862b04Ac198a	True
StableDebtToken	0x3A8ea541597D74ACB33F94533D731940AF516031	True
SupplyLogic	0x55fA0fC04500D04ea7fAe122ae4603b937D8E5A2	**False**
USDC-AToken	0x016341e6Da8da66b33Fd32189328c102f32Da7CC	**False**
USDC-StableDebtToken	0x5faC4FD2e4bCE392d34600d94Aa1114274e54Dff	**False**
USDC-VariableDebtToken	0xE60E1953aF56Db378184997cab20731d17c65004	**False**
USDT-AToken	0x9ca4806fa54984Bf5dA4E280b7AA8bB821D21505	**False**
USDT-StableDebtToken	0x6F977fD05962d67Eb7B16b15684fbEa0462F442d	**False**
USDT-VariableDebtToken	0xa333c6FF89525939271E796FbDe2a2D9A970F831	**False**
VariableDebtToken	0xA48aCc9847Cc1dD2caDA05151C9A78Ba47a305Cb	True
WETH-AToken	0x9002ecb8a06060e3b56669c6B8F18E1c3b119914	**False**
WETH-StableDebtToken	0x9c9158BFF47342A20b7D2Ac09F89e96F3A209b9B	**False**
WETH-VariableDebtToken	0x56f58d9BE10929CdA709c4134eF7343D73B080Cf	**False**

Periphery contracts

contract	address	verified
EmissionManager	0x72D2aB433526d32e6Ee52c03d1562A9E79bf0F19	True
IncentivesV2-Implementation *Proxy*	0x86bd524C09508df7B4B9027464975351B1BC2c92 0x54AB34aB3C723bD2674c7082aA6fFcdfd3A5BEdc	True
Treasury-Controller	0x677C3Cae4F23142c6A8480694554751B462d7326	**False**
Treasury-Implementation *Proxy*	0xC59971Ff27806629D9935fbFBBFC2236961f82C8 0xE52540DBD350c611A1B9c51E97e2A6bc16c09133	**False**
UiIncentiveDataProviderV3	0x91ccF57c1E9A7F5A9537eE59306faF8dA3b7e960	True
UiPoolDataProviderV3	0x8FE0ac76b634B7D343Bd32282B98E9f271B43367	True
WalletBalanceProvider	0xdeEa10da04D867e3303AB6E50FA26C2d8a5e9f70	True
WrappedTokenGatewayV3	0x767b4A087c11d7581Ac95eaFfc1FeBFA26bad3d2	True

Pyth oracles

contract	address	verified
USDC-USD PythNetworkAggregatorV3	0x75D018f04f9cb37936530F7e3A909474565A2467	True
USDT-USD PythNetworkAggregatorV3	0xCf58E8e67F2BcDd977e61bB6FDC1B0EEd6E1939d	True
WETH-USD PythNetworkAggregatorV3	0x517F9cd13fE63e698d0466ad854cDba5592eeA73	True

Table of contents

Findings summary

Storage findings

Core contracts

contract	storage issues initial check
AaveOracle-zkSync	
ACLManager-zkSync	
AToken-zkSync	
DelegationAwareAToken-zkSync	
IncentivesProxy	
Pool-Implementation	
Pool-Proxy-zkSync	
PoolConfigurator-Implementation	
PoolConfigurator-Proxy-zkSync	
PoolDataProvider-zkSync	
ReservesSetupHelper	
ReserveStrategy-rateStrategyStableOne	
ReserveStrategy-rateStrategyStableTwo	
ReserveStrategy-rateStrategyVolatileOne	
StableDebtToken-zkSync	
VariableDebtToken-zkSync	
PoolAddressesProvider-zkSync	

Periphery contracts

contract	storage issues initial check
EmissionManager	
IncentivesV2-Implementation	
UiIncentiveDataProviderV3	
UiPoolDataProviderV3	
WalletBalanceProvider	
WrappedTokenGatewayV3	

Pyth oracles

contract	storage issues initial check
USDC-USD	
WETH-USD	
USDT-USD	

Source code findings

The Mundus team has found no issues concerning the consistency of the code base among **verified** contracts. The major issue is the fact that 21/47 contracts are not verified.

Deployment check: source code

This analysis aims to identify any differences or inconsistencies in the source code of the smart contracts. We perform the analysis in three steps:

1. Analyzing for inconsistency between source code files across deployed smart contracts (excluding well-known dependencies such as OpenZeppelin or Uniswap).
2. Looking for the original commit in the client's repository, which represents all source code of deployed smart contracts in the case of providing the client's git
3. Analyzing the dependencies of the contracts

See number of files statistics in section A1.

Inconsistency between the same project files across contracts (excluding dependencies)

The goal is to check for any differences and inconsistencies in the source code of the same parts of the contracts. We compare each pair of smart contracts in the scope of work (SoW). Files with the same name and relative path included (imported) in both contracts should have the same content.

Summary

The team has found no inconsistencies among **verified** contracts' files.

Searching for the original commit in the client's repository

At this stage, we are looking for the original commit in the client's repository. In the best case, all contracts should be deployed from a single codebase revision to decrease the probability of inconsistency in the contract logic.

core-contracts

contracts	commit	# contracts

AaveOracle-zkSync
ACLManager-zkSync
AToken-zkSync
DelegationAwareAToken-zkSync
IncentivesProxy
Pool-Implementation
+ proxy
PoolConfigurator-Implementation
+ proxy
PoolDataProvider-zkSync
ReservesSetupHelper
ReserveStrategy-rateStrategyStableOne
ReserveStrategy-rateStrategyStableTwo
ReserveStrategy-rateStrategyVolatileOne
StableDebtToken-zkSync
VariableDebtToken-zkSync
PoolAddressesProvider-zkSync
	latest (2023-07-15T02:02:21+05:30):
	93060102ad91e7c9aab45e905e37988261f3f788
	earliest (2023-07-15T01:36:18+05:30):
	2448f46b6b472ba0f83a615f68aa8614866a8321
		17

periphery-contracts

contracts	commit	# contracts

EmissionManager
IncentivesV2-Implementation
UiIncentiveDataProviderV3
UiPoolDataProviderV3
WalletBalanceProvider
WrappedTokenGatewayV3
	latest (2023-07-15T03:13:10+05:30):
	d785e0de52395b7789e0aea9c8a2a14919333af8
	earliest (2023-07-15T03:04:28+05:30):
	841be584a2bae05851da73e3b0984a1c3a804fa9
		6

pyth-oracles

contracts	commit	# contracts

USDC-USD
WETH-USD
USDT-USD
	latest (2023-07-20T00:43:09+05:30):
	f00726842c0006106739b7da8011367329c9db79
	earliest (2023-07-17T02:11:39+05:30):
	806d83aa0171dba957652cac521c738289c3441c
		3

Summary

All **verified** contracts fall under single commit in their respective repositories.

Analyzing the dependencies of the contracts

The goal is to check the consistency of every dependency version and identify any changes across every dependency codebase.

Periphery contracts

contract	core-contracts
EmissionManager	93060102ad91e7c9aab45e905e37988261f3f788
IncentivesV2-Implementation	93060102ad91e7c9aab45e905e37988261f3f788
UiIncentiveDataProviderV3	93060102ad91e7c9aab45e905e37988261f3f788
UiPoolDataProviderV3	93060102ad91e7c9aab45e905e37988261f3f788
WalletBalanceProvider	93060102ad91e7c9aab45e905e37988261f3f788
WrappedTokenGatewayV3	93060102ad91e7c9aab45e905e37988261f3f788

Pyth oracles

contract	@pythnetwork
USDC-USD	v2.2.0
WETH-USD	v2.2.0
USDT-USD	v2.2.0

Summary

All **verified** contracts use consistent versions of respective dependencies.

Deployment check: storage

We thoroughly examine both public and private storage, as well as immutable and constant variables, to ensure that there are no misconfigurations, especially:

1. Incorrect or outdated addresses to other smart contracts referenced in the scope of work (SoW) - this includes addresses stored in variables, mappings, and other data structures.
2. Any references to other smart contracts or externally owned accounts (EOAs) that may be incorrect or outdated.
3. Any incorrect protocol settings stored in variables or other data structures.
4. Misconfigurations related to the roles and permissions of the contract.
5. Governance issues that may impact the operation and business logic of the smart contract.

WIP

Disclaimers

Mundus disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical disclaimers

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

Appendix

A1. Statistics among verified contracts

Core contracts

contract	# files
AaveOracle-zkSync	7
ACLManager-zkSync	10
AToken-zkSync	21
DelegationAwareAToken-zkSync	23
Pool-Implementation	46
Pool-Proxy-zkSync	6
PoolConfigurator-Implementation	22
PoolConfigurator-Proxy-zkSync	6
PoolDataProvider-zkSync	16
ReservesSetupHelper	25
ReserveStrategy-rateStrategyStableOne	9
ReserveStrategy-rateStrategyStableTwo	9
ReserveStrategy-rateStrategyVolatileOne	9
StableDebtToken-zkSync	20
VariableDebtToken-zkSync	22
PoolAddressesProvider-zkSync	10

Periphery contracts

contract	# files
EmissionManager	9
IncentivesV2-Implementation	12
IncentivesProxy	6
UiIncentiveDataProviderV3	21
UiPoolDataProviderV3	28
WalletBalanceProvider	9
WrappedTokenGatewayV3	18

Pyth oracles

contract	# project files	# @pythnetwork files
USDC-USD	1	3
WETH-USD	1	3
USDT-USD	1	3