Lendefi

Protocol

SMART CONTRACT AUDIT

10.04.2022

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer 4
2. About the Project and Company 5
2.1 Project Overview 6
3. Vulnerability & Risk Level 7
4. Auditing Strategy and Techniques Applied 8
4.1 Methodology 8
4.2 Used Code from other Frameworks/Smart Contracts 9
4.3 Metrics / CallGraph 10
4.4 Metrics / Source Lines & Risk 11
4.5 Metrics / Capabilities 12
4.6 Metrics / Source Unites in Scope 13
4.7 Inheritance Graph 24
4.8 Fork details 25
5. Scope of Work 26
5.1 Findings Overview 27
5.2 Manual and Automated Vulnerability Test 28
5.2.1 Wrong import of OpenZeppelin library 28
5.2.2 Different pragma versions identified 29
5.2.3 Missing natspec documentation 30
5.2.4 Remove unused code 30
5.2.5 Rename bzx branding 31
5.2.6 Error or require messages are not clear 32

5.2.7 Constant name must be capitalized 32
5.2.8 Merge commit from original source 34
5.3 SWC Attacks 35
5.4 Associated audits with the forked codebase 39
6. Executive Summary 40
7. Deployed Smart Contract 40
8. About the Auditor 42

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of DOGON SIRIUS LIMITED (Lendefi Protocol). If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (23.06.2021)	Layout
0.4 (26.06.2021)	Automated Security Testing Manual Security Testing
0.5 (30.06.2021)	Verify Claims and Test Deployment
0.6 (01.07.2021)	Testing SWC Checks
0.9 (02.07.2021)	Summary and Recommendation
1.1 (04.07.2021)	Final document

2. About the Project and Company

Company address:
DOGON SIRIUS LIMITED
Unit 3A-16, Level 3A, Labuan Times Square
Jalan Merdeka
87000 Labuan, Malaysia

Website: https://www.lendefi.finance

Twitter: https://twitter.lendefi.finance

Telegram: https://telegram.lendefi.finance

Medium: https://medium.lendefi.finance

GitHub: https://github.lendefi.finance

LinkedIn: https://linkedin.lendefi.finance

Facebook: https://facebook.lendefi.finance

Instagram: https://www.instagram.com/lendefi.finance

2.1 Project Overview

The Lendefi protocol (the “Protocol”) allows secured lending, giving the much-needed confidence to the lenders in a highly volatile crypto market. Secure lending options will open up lending opportunities for traditional and private lenders to access higher interest rates without getting direct exposure to the crypto market fluctuations.

Lendefi protocol cuts the middle-man out of the lending process and eliminates the red tape involved with the lending and borrowing. This removes any counterparty risk between the borrower and the lender, who then can deal on a trustless basis. The lender will receive a variable interest and be secured by the liquidity provided on the DeFi ecosystem in such protocols as Uniswap . Hence, if the borrower is not able to maintain their loan, the Protocol will ensure the lender is repaid and the borrower credited with the remaining equity. Borrowers can select from a wide variety of supported assets to invest by borrowing funds from the Protocol.

Supported assets can be added and removed via Lendefi’s decentralized governance mechanism (the “DAO”). The base currency for lending and borrowing is USDC, hence making it more user-friendly and fostering mainstream adoption. Lendefi has specifically chosen USDC because it is the safest stable coin from a custody and reputation perspective, given it is a collaboration between Coinbase and Circle, and undergoes regular audits and is subject to regulatory compliance.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

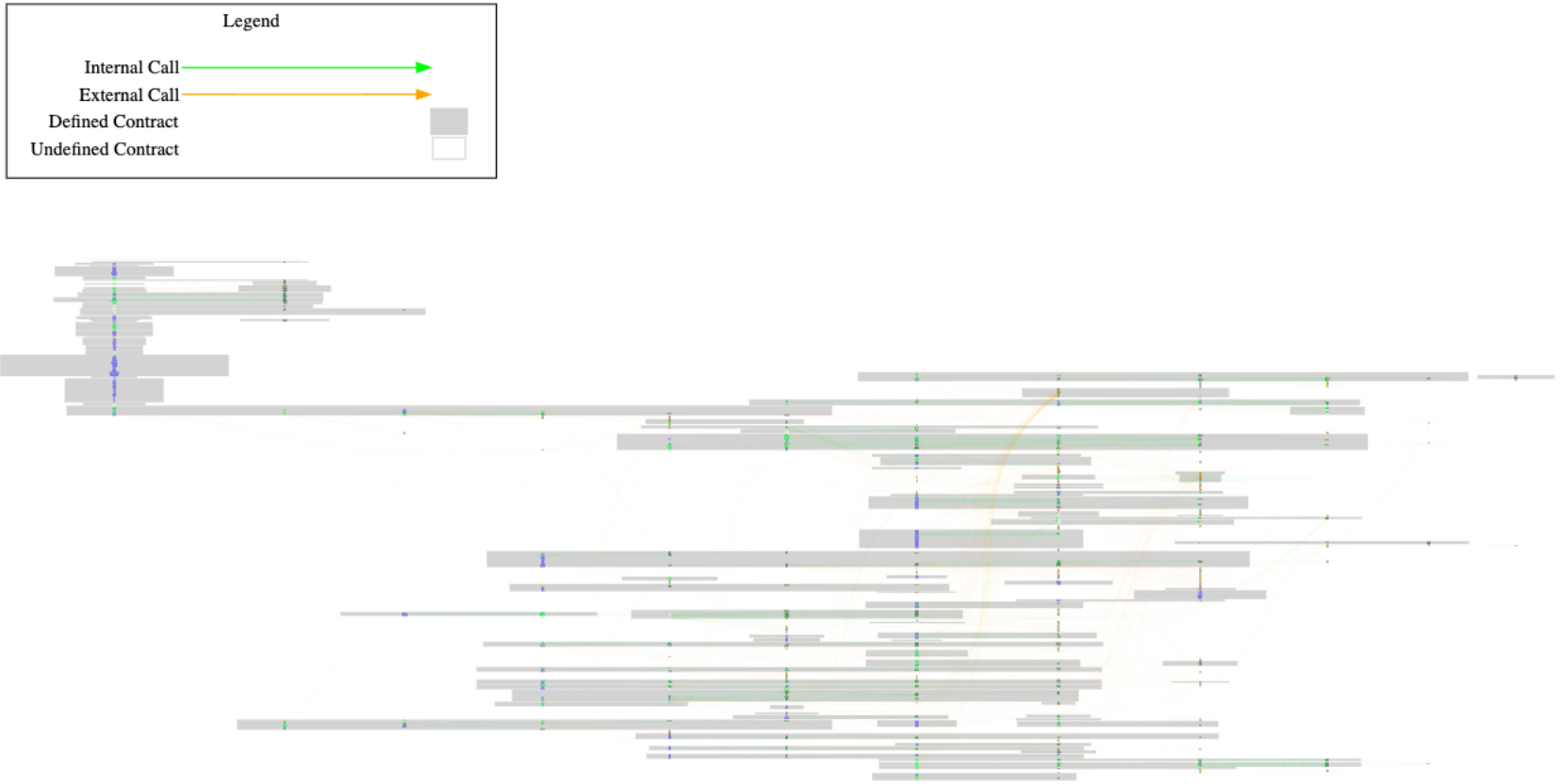
The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

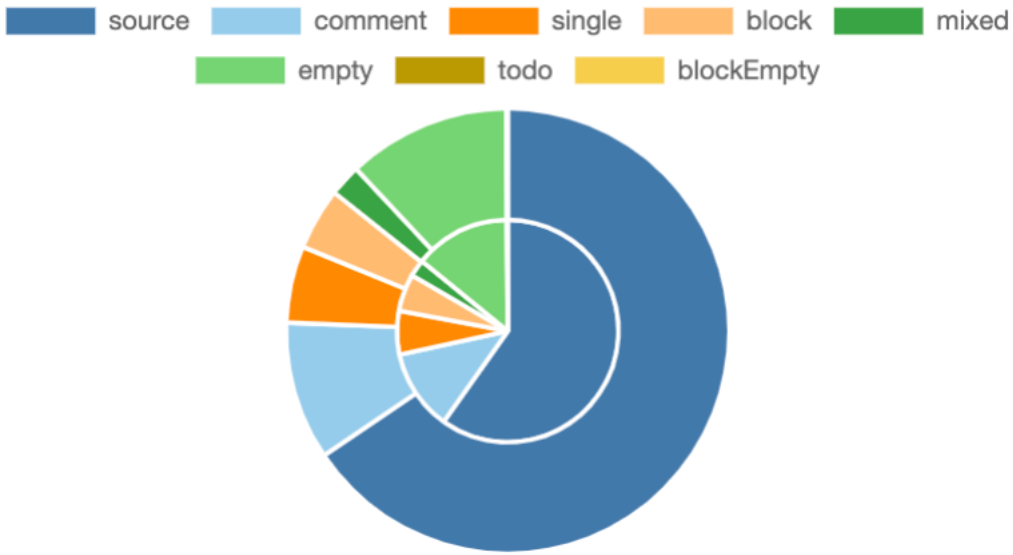
Dependency / Import Path	Source
@openzeppelin/contracts/access/Context.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/utils/Context.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/access/Ownable.sol
@openzeppelin/contracts/util/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/math/SafeMath.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/util/Address.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/utils/Address.sol

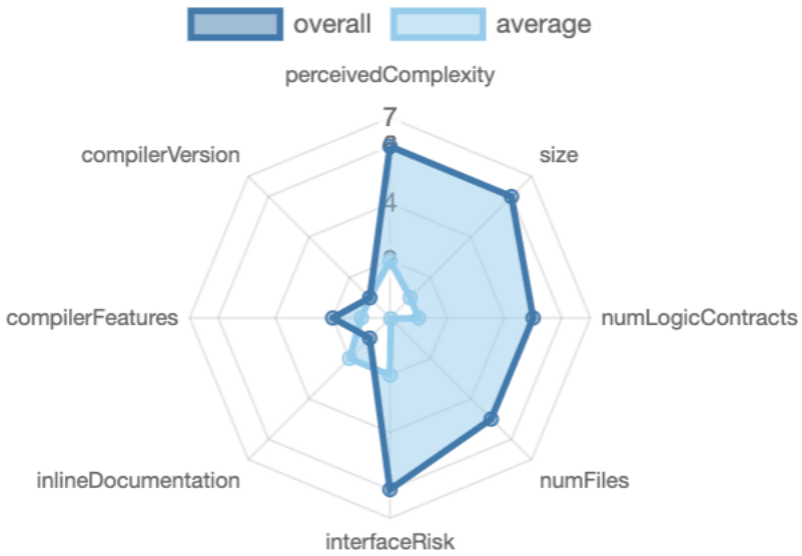
4.3 Metrics / CallGraph



See full version: http://chainsulting.de/wp-content/uploads/2021/07/solidity-metrics-lendefi.html

4.4 Metrics / Source Lines & Risk





4.5 Metrics / Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

`>=0.5.0 <0.6.0`
`0.5.17`
`0.5.16`
`0.6.12`
`^0.7.6`
`>=0.6.0 <0.7.0`
	`ABIEncoderV2`	`yes`	`yes`
			(50 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`	`yes`	`yes`	`yes`	`yes`	

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Public	Payable
624	38

External	Internal	Private	Pure	View
341	642	2	40	318

Total	Public
338	272

4.6 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/interfaces/IVestingToken.sol	1		26	12	4	4	9	
📝🔍	contracts/interfaces/IChai.sol	1	1	50	13	5	4	18	
🔍	contracts/interfaces/IPancakePair.sol		1	59	14	5	4	55	
🔍	contracts/interfaces/IWeth.sol		1	12	10	3	4	8	💰
🔍	contracts/interfaces/IPancakeRouter02.sol		1	212	11	3	5	61	💰
🔍	contracts/interfaces/IUniswapV2Router.sol		1	45	11	3	8	9	
📝	contracts/interfaces/IWethERC20.sol	1		12	12	4	4	5	
🔍	contracts/interfaces/IWBNB.sol		1	24	9	3	4	13	
🔍	contracts/interfaces/IPancakeFactory.sol		1	23	12	4	4	17	
🔍	contracts/interfaces/ILoanPool.sol		1	30	10	3	4	9	
📝	contracts/interfaces/IERC20.sol	1		21	13	6	4	16	
📝	contracts/openzeppelin/Ownable.sol	1		68	68	29	28	21	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📚	contracts/openzeppelin/SafeERC20.sol	1		77	77	36	29	29	
📝	contracts/openzeppelin/ReentrancyGuard.sol	1		40	40	12	20	5	
📝	contracts/openzeppelin/Context.sol	1		28	28	11	14	1	🔆
📚	contracts/openzeppelin/SignedSafeMath.sol	1		90	90	29	49	9	
📚	contracts/openzeppelin/Address.sol	1		71	71	17	49	14	🖥️📩⚡
📚	contracts/openzeppelin/SafeMath.sol	1		183	183	53	111	14	
📝	contracts/modules/LoanMaintenance/LoanMaintenance.sol	1		835	713	578	54	403	🖥️🧪💰📩🧮
📝🔍	contracts/adapters/TokenRegistry.sol	1	1	60	42	20	20	28	🧪
📝	contracts/modules/SwapsExternal/SwapsExternal.sol	1		156	109	89	11	53	🧪💰📩
📝	contracts/events/FeesEvents.sol	1		52	52	39	4	1	
📝	contracts/events/LoanSettingsEvents.sol	1		38	38	29	4	1	
📝	contracts/events/LoanClosingsEvents.sol	1		68	68	56	5	1	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/events/LoanMaintenanceEvents.sol	1		80	80	65	21	1	
📝	contracts/events/SwapsEvents.sol	1		27	27	18	4	1	
📝	contracts/events/LoanOpeningsEvents.sol	1		52	52	41	5	1	
📝	contracts/events/ProtocolSettingsEvents.sol	1		106	106	84	4	1	
📝	contracts/modules/LoanClosings/LoanClosingsBase.sol	1		976	838	679	97	263	🖥️🧪📤🧱
📝	contracts/proxies/05/*Upgradeable*0_5.sol	1		13	13	5	4	4	
📝	contracts/modules/ProtocolSettings/ProtocolSettings.sol	1		500	400	334	14	278	🖥️🧪📤👥🧱
📝	contracts/modules/LoanClosings/LoanClosingsWithGasToken.sol	1		119	61	45	14	40	🧪💰
📝	contracts/proxies/05/*Proxy*05.sol	1		42	42	28	4	65	🖥️💰👥
📝	contracts/modules/LoanClosings/LoanClosings.sol	1		110	60	44	14	34	🧪💰
📝	contracts/ArbitraryCaller.sol	1		26	21	13	4	38	🖥️💰📤⚡
📝	contracts/modules/_TestnetOnly/TmpAdminClosings.sol	1		158	114	91	10	68	💰

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/modules/_TestnetOnly/TmpAdmin.sol	1		128	116	93	4	69	🖥️🧪
📝	contracts/modules/_TestnetOnly/TmpAdminInterestSettlement.sol	1		96	86	63	7	47	🖥️🧪
📝	contracts/modules/LoanSettings/LoanSettings.sol	1		169	136	110	6	105	🖥️🧪🧮
📝	contracts/modules/LoanOpenings/LoanOpenings.sol	1		707	560	455	100	181	🧪💰🧮
🔍	contracts/swaps/ISwapsImpl.sol		1	32	10	3	4	7	
📝	contracts/swaps/SwapsUser.sol	1		231	194	146	36	55	👥
📝	contracts/connectors/DAppHelper.sol	5		122	67	47	11	163	🧪🔆
📝	contracts/core/Objects.sol	1		21	21	13	4	11	
📝	contracts/feeds/DollarPegFeed.sol	1		27	23	9	9	8	
📝	contracts/mixins/VaultController.sol	1		137	108	89	4	38	📤
📝	contracts/feeds/PriceFeeds_POLYGON.sol	1		390	283	223	15	132	
📝	contracts/core/Constants.sol	1		52	52	13	28	12	
🔍	contracts/feeds/IPriceFeedsExt.sol		1	11	10	3	4	3	🔆

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/feeds/AAVEToLENDFeed.sol	1		22	18	7	8	7	
📝	contracts/swaps/connectors/SwapsImplUniswapV2_BSC.sol	1		311	256	209	20	181	🖥️
📝	contracts/feeds/PriceFeeds.sol	1		398	291	229	16	140	
📝	contracts/protocoltoken/BZRXv1Converter.sol	1		88	73	47	11	32	📥
📝	contracts/feeds/AAVEToUSD_POLYGON.sol	1		26	22	11	5	11	
📚	contracts/protocoltoken/Checkpointing.sol	1		152	125	70	39	18	
📝	contracts/protocoltoken/VBZRXWrapper.sol	1		170	170	128	16	88	📥
📝	contracts/mixins/FeesHelper.sol	1		219	178	135	21	68	🖥️🧱
📝	contracts/protocoltoken/CheckpointingToken.sol	1		181	112	82	7	38	
🔍	contracts/feeds/IPriceFeeds.sol		1	92	10	3	4	21	
📝	contracts/core/State.sol	1		93	89	55	42	51	
📝	contracts/mixins/InterestUser.sol	1		77	68	50	5	21	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/connectors/loantoken/LoanTokenBase.sol	1		48	48	30	7	27	
📝	contracts/protocoltoken/BZRXVestingToken.sol	1		275	217	150	32	107	📩
📝	contracts/connectors/loantoken/LoanTokenLogicWeth.sol	1		110	91	62	15	38	🧪💰📩
📝	contracts/protocoltoken/VBZRXWrapper_alt.sol	1		196	196	146	12	98	📩
📝	contracts/protocoltoken/BZRXToken.sol	1		34	30	18	5	15	
📝	contracts/protocoltoken/iETHBuyBack.sol	1		146	117	75	16	57	📩
📝	contracts/staking/StakingUpgradeable.sol	1		13	13	5	4	4	
📝	contracts/protocoltoken/TraderCompensation.sol	1		174	132	93	12	68	📩
📝	contracts/connectors/loantoken/Pausable.sol	1		31	26	15	5	12	🖥️🧱
📝🔍	contracts/protocoltoken/iETHBuyBackV2.sol	1	1	237	172	118	19	106	📩
📚	contracts/mixins/EnumerableBytes32Set.sol	1		209	173	82	73	45	🖥️

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/core/Protocol.sol	1		68	54	38	5	80	🖥️🧪💰👥🧱
📝	contracts/mixins/LiquidationHelper.sol	1		69	59	44	7	21	
📝	contracts/feeds/PriceFeeds_BSC.sol	1		392	285	223	16	132	
📝	contracts/feeds/FixedPriceFeed.sol	1		24	20	9	6	9	
📝	contracts/connectors/loantoken/LoanTokenLogicStandard.sol	1		1363	1014	764	141	496	🖥️🧪💰📤⚡🧱
📝	contracts/staking/StakingProxy.sol	1		51	47	34	4	62	🖥️💰👥
📝	contracts/connectors/loantoken/AdvancedTokenStorage.sol	1		77	62	45	4	12	
📝	contracts/staking/StakingState.sol	1		60	60	39	14	34	
📝	contracts/staking/StakingV1.sol	1		1245	1020	754	141	456	🧪📤
📝	contracts/connectors/loantoken/LoanTokenSettings.sol	1		122	90	64	7	61	🖥️🧪📤⚡
📝	contracts/core/objects/LoanParamsStruct.sol	1		20	20	13	12	1	
📝	contracts/staking/StakingConstants.sol	1		126	126	75	36	29	
📝	contracts/core/objects/LoanStruct.sol	1		24	24	17	16	1	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/core/objects/LenderInterestStruct.sol	1		17	17	10	9	1	
📝	contracts/core/objects/OrderStruct.sol	1		18	18	11	10	1	
📝	contracts/farm/GovToken.sol	1		25	25	16	2	21	
📝	contracts/farm/Proxy.sol	1		44	44	32	1	60	🖥️💰👥
📝	contracts/connectors/loantoken/TestnetOnly/LogicMock.sol	1		102	88	63	10	34	
📝🔍	contracts/farm/MintCoordinator_Polygon.sol	1	1	50	44	28	5	39	
📝	contracts/farm/BGovToken.sol	1		241	211	130	48	81	🖥️🧮🧹
📝	contracts/farm/MasterChef_Polygon.sol	1		505	460	344	49	286	🧪🔷
📝	contracts/farm/FeeExtractAndDistribute_BSC.sol	1		284	245	178	27	128	🧪🔷
📝	contracts/connectors/loantoken/LoanToken.sol	1		61	54	39	5	65	🖥️💰👥
📝	contracts/core/objects/LoanInterestStruct.sol	1		15	15	8	7	1	
📝🔍	contracts/farm/MintCoordinator_BSC.sol	1	1	48	43	22	13	28	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/farm/MasterChef_BSC.sol	1		429	389	292	50	264	🧪📤
📝	contracts/connectors/gastoken/TokenHolder.sol	2		139	96	73	9	65	
📝	contracts/connectors/loantoken/AdvancedToken.sol	1		103	74	51	5	28	
📝	contracts/connectors/gastoken/GasTokenUser.sol	2		70	60	35	14	26	
🔍	contracts/connectors/loantoken/interfaces/ProtocolSettingsLike.sol		1	21	13	5	4	5	🧪
📝	contracts/connectors/loantoken/LoanTokenLogicDai.sol	1		561	423	307	54	357	🖥️🧪💰📤⚡🧮
📝	contracts/connectors/loantoken/LoanTokenSettingsLowerAdmin.sol	1		127	102	70	19	85	🖥️🧪🧮
🔍	contracts/staking/interfaces/IBZxPartial.sol		1	35	17	9	4	7	
📝	contracts/connectors/loantoken/interfaces/ProtocolLike.sol	1		118	36	9	22	26	🧪💰
🔍	contracts/connectors/loantoken/interfaces/FeedsLike.sol		1	16	10	3	4	3	
🔍	contracts/helpers/IToken.sol		1	22	11	3	5	13	

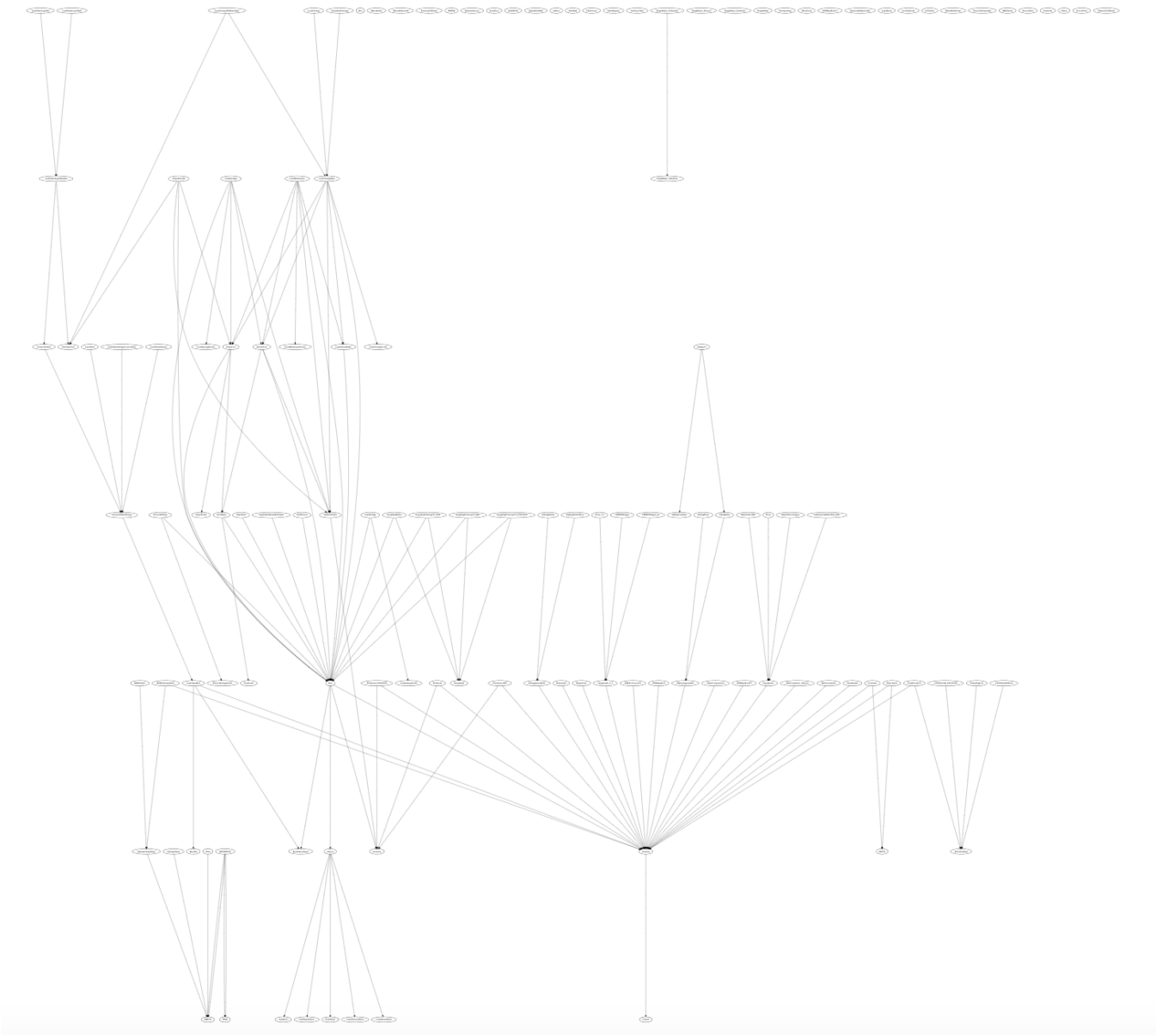
Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/staking/interim/StakingInterimProxy.sol	1		51	47	34	4	62	🖥🧪💰👥
📝	contracts/helpers/HelperImpl.sol	1		106	78	61	5	174	🧪
🔍	contracts/staking/interfaces/ICurve3Pool.sol		1	19	10	3	4	5	
📝	contracts/staking/interim/StakingInterimState.sol	1		48	48	31	12	26	
📝	contracts/swaps/connectors/SwapsImplUniswapV2_POLYGON.sol	1		340	285	243	12	195	🖥
📝	contracts/swaps/connectors/SwapsImplKyber.sol	1		170	140	109	15	76	🖥
📝	contracts/staking/interim/StakingInterim.sol	1		475	362	290	10	200	🖥🧪
📝	contracts/helpers/HelperProxy.sol	1		52	52	33	5	61	🖥💰👥
🔍	contracts/farm/interfaces/IWethERC20.sol		1	12	12	4	4	5	🔆
📝	contracts/farm/interfaces/Upgradeable.sol	1		13	13	5	4	4	
🔍	contracts/farm/interfaces/IUniswapV2Router.sol		1	45	11	3	8	9	
📝	contracts/swaps/connectors/SwapsImplUniswapV2_ETH.sol	1		320	265	226	11	188	🖥

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/farm/interfaces/IPriceFeeds.sol		1	24	10	3	4	5	
🔍	contracts/farm/interfaces/IMasterChefPartial.sol		1	13	10	3	4	3	
🔍	contracts/farm/interfaces/ICurve3Pool.sol		1	19	10	3	4	5	
🔍	contracts/farm/interfaces/IWeth.sol		1	12	10	3	4	8	💰
🔍	contracts/farm/interfaces/IBZxPartial.sol		1	35	17	9	4	7	
📝📚🔍	**Totals**	**109**	**27**	**18360**	**14569**	**10495**	**2167**	**7729**	🖥️🧪💰📩⚡👥🧮🔮🔆

Legend: [━]

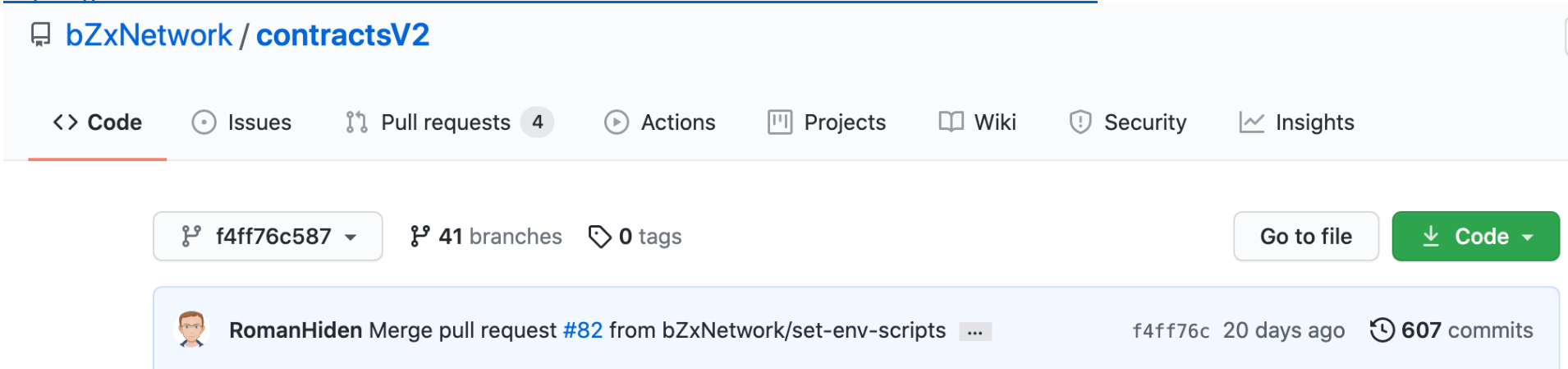
- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

4.7 Inheritance Graph



4.8 Fork details

Last Commit (f4ff76c587) from original source:
https://github.com/bZxNetwork/contractsV2/tree/f4ff76c587dfadc25fd12a88c8feeac9ec510242



5. Scope of Work

The Lendefi Team provided us with the files that needs to be tested. The scope of the audit are the Lendefi Protocol contracts.

Following contracts has been tested:

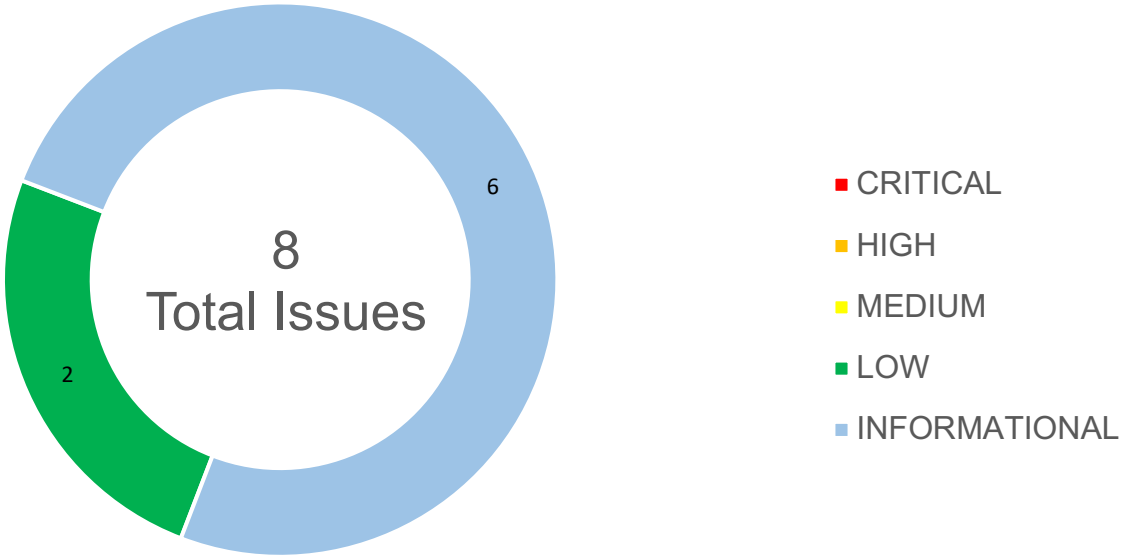
- https://github.com/lendefi/contracts/tree/3002e7c43da3b075146c274b15d184b511b69474

The team put forward the following assumptions regarding the security, usage of the contracts:

- The changes that have been made to the forked bZx Protocol are not affecting the overall security
- The smart contract is coded according to the newest standards and in a secure way
- The changes that the bZx Protocol Team has made, after the last audit in September 2020, are not affecting the overall security

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

5.1 Findings Overview



No	Title	Severity	Status
5.2.1	Wrong import of OpenZeppelin library	LOW	OPEN
5.2.2	Different pragma versions identified	LOW	OPEN
5.2.3	Missing natspec documentation	INFORMATIONAL	OPEN
5.2.4	Remove unused code	INFORMATIONAL	OPEN
5.2.5	Rename bzx branding	INFORMATIONAL	OPEN
5.2.6	Constant name must be capitalized	INFORMATIONAL	OPEN
5.2.7	Improper Session Management	INFORMATIONAL	OPEN

5.2.8	Merge commit from original source	INFORMATIONAL	

5.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.2.1 Wrong import of OpenZeppelin library
Severity: LOW
Status: ACKNOWLEDGED
File(s) affected: TokenHolder.sol, LoanTokenBase.sol, FastGasFeedMock.sol, TmpAdmin.sol, TmpAdminInterestSettlement.sol, ProtocolSettings.sol, Proxy0_5.sol, Upgradeable_0_5.sol, StakingInterimState.sol, SwapImplKyber.sol,

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation, some OpenZeppelin files are part of the repository. This violates OpenZeppelin's MIT license, which requires the	`SafeMath, Ownable, SafeERC20, Address, ReentrancyGuard, SignedSafeMath`	We highly recommend using npm (import "@openzeppelin/contracts/..) in order to guarantee that original OpenZeppelin contracts are used with no modifications. This also allows for any bug-fixes to be easily integrated into the codebase.

license and copyright notice to be included if its code is used. Moreover, updating code manually is error-prone.		https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/

5.2.2 Different pragma versions identified

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation, several pragma versions have been identified, which can lead to inconsistency and further problems while deployment.	>=0.5.0 <0.6.0 0.5.17 0.5.16 0.6.12 ^0.7.6 >=0.6.0 <0.7.0	It is recommended to normalize all files to one consistent pragma version. ex. 0.5.17 (Most used version)

INFORMATIONAL ISSUES

5.2.3 Missing natspec documentation

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: LoanTokenLogicStandard.sol, LoanTokenSettings.sol,, FeesEvents.sol, LoanClosingEvents.sol, LoanOpeningsEvents.sol, LoanSettingsEvents.sol, ProtocolSettingsEvents.sol, SwapsEvents.sol, MasterChef_BSC.sol, IPriceFeeds.sol, PriceFeeds_BSC.sol,

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

5.2.4 Remove unused code

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: TokenRegistry.sol, LoanTokenLogicDai.sol, LoanTokenLogicStandard.sol, Constants.sol, StakingV1.sol

Attack / Description	Code Snippet	Result/Recommendation
The chances that the unused code will ever be used again is very unlikely.	Line: 29 – 31 and more (TokenRegistry.sol)	It is recommended to remove unused code for a better readability.

	`//address public constant bZxContract = 0xD8Ee69652E4e4838f2531732a46d1f7F584F0b7f; // mainnet`
	`//address public constant bZxContract = 0x5cfba2639a3db0D9Cc264Aa27B2E6d134EeA486a; // kovan`
	`//address public constant bZxContract = 0xC47812857A74425e2039b57891a3DFcF51602d5d; // bsc mainnet`	

5.2.5 Rename bzx branding

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
There are still brandings from bzx included, which would lead to confusion.	NA	It is recommended to rename files / functions / variables and keep the branding consistent to lendefi or LDFI. You must keep the Apache 2.0 License header. For more information https://fossa.com/blog/open-source-licenses-101-apache-license-2-0/

5.2.6 Error or require messages are not clear

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: MasterChef_BSC.sol

Attack / Description	Code Snippet	Result/Recommendation

User won't understand the error messages or correctly interpret it while debugging.	Line 352 (MasterChef_BSC.sol)
	`require(user.amount >= _amount, "withdraw: not good");`
	
	Line 387 (MasterChef_BSC.sol)
	`require(msg.sender == devaddr, "dev: wut?");`
		It is recommended to write clear error messsages

5.2.7 Constant name must be capitalized

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: TokenRegistry.sol, GasTokenUser.sol, TokenHolder.sol, LoanTokenLogicStandard.sol, LoanTokenSettings.sol, Pausable.sol, Constants.sol, FeeExtractAndDistribute_BSC.sol, MasterChef_BSC.sol, BZRXToken.sol,

Attack / Description	Code Snippet	Result/Recommendation
Linter: Constant name must be in capitalized SNAKE_CASE [const-name-snakecase]	Line: 32 (TokenRegistry.sol) `address public constant bZxContract = 0xe638C33001a7e403F560946ff3279F9B07b6AD29; // bsc testnet`	Use capital letters for constants with underscore to separate words like MAX_BLOCKS.

	Line: 21 / 25 (GasTokenUser.sol)
	`ITokenHolderLike constant public gasToken = ITokenHolderLike(0xb628bbB1BA814f37448793475D48aaFa0AdBA97d); // bsc testnet`
	
	`ITokenHolderLike constant public tokenHolder = ITokenHolderLike(0x662063E98F3e1276798e8C6af6Bc5B03D92f3b40); // bsc testnet`
	
	Line 25 (TokenHolder.sol)
	`IChiToken constant public gasToken = IChiToken(0xb628bbB1BA814f37448793475D48aaFa0AdBA97d); // bsc testnet`
	
	Line 22 (LoanTokenBase.sol)
	`int256 internal constant sWEI_PRECISION = 10**18;`
	
	Line 13 (Pausable.sol)
	`bytes32 internal constant Pausable_FunctionPause = 0xa7143c84d793a15503da6f19bf9119a2dac94448ca45d77c8bf08f57b2e91047;`
	
	Line 19 – 20 (Constants.sol)
	`string internal constant UserRewardsID = "UserRewards";`
	`string internal constant LoanDepositValueID = "LoanDepositValue";`
	
	Line 79 (MasterChefBSC.sol)
	`MintCoordinator public constant coordinator = MintCoordinator(0xf524fbAC6288fb8cdf2A32dB7e1fd28b4a57807F); // bsc testnet`
	
	Line 13 – 17 (BZRXToken.sol)
	`string public constant name = "Lendefi Protocol Token";`
		https://www.tutorialspoint.com/solidity/solidity_style_guide.htm

	`string public constant symbol = "LDFI";`
	`uint8 public constant decimals = 18;`
	
	`uint256 internal constant totalSupply_ = 1030000000e18; // 1,030,000,000 BZRX`	

5.2.8 Merge commit from original source

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: NA

Attack / Description	Code Snippet	Result/Recommendation
Some useful commits from the original source are not merged.	https://github.com/bZxNetwork/contractsV2/commits/development	Merging the commits and monitoring from the original source would actively increase the forked versions quality.

5.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	X
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	X
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	X
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

5.4 Associated audits with the forked codebase

Certik	https://bzx.network/pdfs/bZx_v2_Audit%E2%80%93Report_CertiK.pdf	Sep. 2020
PeckShield	https://bzx.network/pdfs/peckshield-audit-report-bZxV2-v1.0rc1.pdf	Sep. 2020
Certik	https://bzx.network/pdfs/BZRX_vBZRX_CertiK_Verification_Report_1_07_11_2020.pdf	July. 2020
Certik	https://bzx.network/pdfs/BZRX_vBZRX_CertiK_Report_1_07_11_2020.pdf	July. 2020
Certik	https://bzx.network/pdfs/CertiK%20Verification%20Report%20for%20bZx.pdf	March. 2020
Certik	https://bzx.network/pdfs/CertiK%20Verification%20Report%20for%20bZx.pdf	February. 2020
ZK Labs	https://github.com/mattdf/audits/blob/master/bZx/bzx-audit.pdf	September 2018

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found, after the manual and automated security testing. Only low and informational issues were found, to increase the code quality. Overall, everything worked as it was supposed to be. As this protocol is actively developed by bZx Team, we would recommend to actively monitor the repo and update the own codebase accordingly.

7. Deployed Smart Contract

VERIFIED

Protocol	0x74e974541A873afbcb2D89E973fb21F0db360182
PriceFeeds_BSC	0xd475ac73B1B5c3cB5b1e468f0f30D76F98026A6F
SwapsImplUniswapV2_BSC	0x998d51380b461bc15A756A31DF036Da7E5E80A01
ProtocolSettings	0x0Bd6171E1080F5a7Ef9CB0FF57b55C53Df2C8A08
ArbitraryCaller	0x72d60aa125996aCDF6118BDD2C4724a505684e1d
TokenRegistry	0xcf02Cc87345D07BB1CA365fF6bf57A8e600F5692
TokenHolder	0x66009c26CbA9765d173F1BEB3B86D18a67276d25
GasTokenUser	0x8863B76173AFA3383a9960899Eee3b9A3E7a662e
LoanTokenLogicStandard	0x74a88135f15046CCE01B0F7B75298dd636ad4e85
LoanTokenSettings	0xB967D536784a42f97eAfA9fBcF5Ea2303f360d56
LoanTokenLogicWeth	0x737C81faC021F54ff2049778F45Dc16b86871338

LoanTokenSettingsLowerAdmin	0x7351cf7A694Ab2656Abbcc9d2B21E00218AdF134
HelperImpl	0x9D55165868bADc80668F4330863b23977DB4C548
DAppHelper	0x0B970914ddE8b54b514f740C8C1168D409A28853
LoanSettings	0x41852D666F9F7e04fb294D037a388A705F505CDD
LoanOpenings	0xad15Cc404e724125B15E89734B560D86a2017C41
LoanMaintenance	0x270ea4455c4AB5dBd15CD54B73213d9c14Fd378d
LoanClosings	0xb71Cb59FA04591559377931a83c95bca8C5971ef
SwapsExternal	0x9eb11D4d403BbbEE7e024F13fc0D637Befe94025
Liquidator1	0x784E1E3AF8f700d211a8428842Ca6D121A25E74c
Liquidator2	0x7B459d77D50522f14DB3ebc7Cc4599cb35D7eF66
iAUTO	0x18a60aeE571e26418ab34c28D26a290ca3558f9F
iCAKE	0x85e136591370A02B48eA9928315AcD210ba64175
iBTC	0x97F17Ab160724A37A24E6fcC1a5da97127489B14
iETH	0x2D506155b3D80368bBdd604DdA4F3A471efB1364
iBNB	0x3814fBBC5726B5815CF0B2c70ee2413f38e32176
iDOT	0xd9Cd58cf5F8b363b147Db821534DD7163D6C01de
iADA	0x31579e2493A0e7779206A58475D2B5f08b90d13F
iLINK	0xb4854941E702ADf54F16665b95FFA3868f51070C
iUNI	0x414525f6AF65ba23a538D68A15aCB09a2FB45563
iBUSD	0xF3aEBedEEE632307F27185d067C105e725ac5e15
iUSDT	0x40211e07dFcB6140eb88A65DfF877d680713cdCc

8. About the Auditor

Chainsulting is a professional software development firm based in Germany that provides comprehensive distributed ledger technology (DLT) solutions. Some of their services include blockchain development, smart contract audits and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Chainsulting currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions tailored to the clients' evolving business needs.

The blockchain security provider brings the highest security standards to crypto and blockchain platforms, helping to foster growth and transparency within the fast-growing ecosystem.

Check our website for further information: https://chainsulting.de