



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
13. December, 2021

For



BaksDAO

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Scope of Work	13
Inheritance Graph	13
Verify Claims	14
Modifiers	19
CallGraph	22
Source Units in Scope	23
Critical issues	24
High issues	24
Medium issues	24
Low issues	24
Informational issues	24
Audit Comments	25
BaksDAO Test Protocol	26
SWC Attacks	28

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	13. December 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.0	14. December 2021	Added Manual testing and GitHub commit

Network

Binance Smart Chain (BEP20)

Website

<https://baksdao.com/>

Telegram

https://t.me/BaksDAO_INT_COMMUNITY

Twitter

<https://twitter.com/BaksDAO>

Facebook

<https://www.facebook.com/BaksDAOint/>

Medium

<https://medium.com/@BaksDAO>

Github

<https://github.com/BaksDAO/contracts-mirror/tree/master/contracts>

Description

A platform to get more from your crypto assets

Project Engagement

During the 7th of December 2021, **BaksDAO Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link

v1.0

• TBA

The BaksDAO Team provided us access to the private GitHub repository. We have audited „contracts-mirror“ with the commit 3650ea088507b4ea8df181aa85dec89a8392887e

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

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.

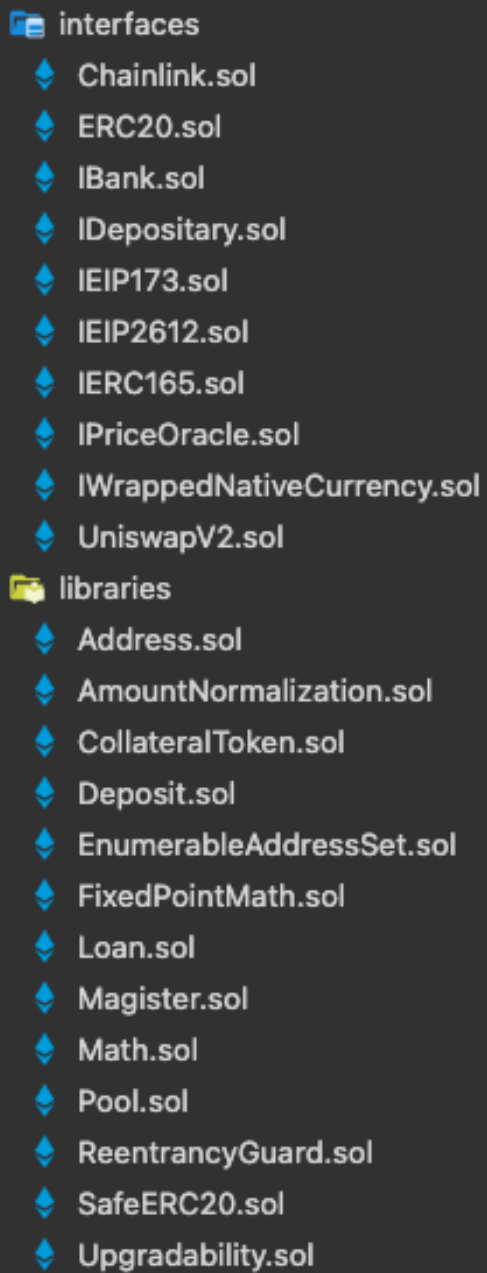
Methodology

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 SolidProof 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 SolidProof 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.

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

Imported packages:



```

📁 interfaces
  🔹 Chainlink.sol
  🔹 ERC20.sol
  🔹 IBank.sol
  🔹 IDepository.sol
  🔹 IEIP173.sol
  🔹 IEIP2612.sol
  🔹 IERC165.sol
  🔹 IPriceOracle.sol
  🔹 IWrappedNativeCurrency.sol
  🔹 UniswapV2.sol
📁 libraries
  🔹 Address.sol
  🔹 AmountNormalization.sol
  🔹 CollateralToken.sol
  🔹 Deposit.sol
  🔹 EnumerableAddressSet.sol
  🔹 FixedPointMath.sol
  🔹 Loan.sol
  🔹 Magister.sol
  🔹 Math.sol
  🔹 Pool.sol
  🔹 ReentrancyGuard.sol
  🔹 SafeERC20.sol
  🔹 Upgradability.sol
```


Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

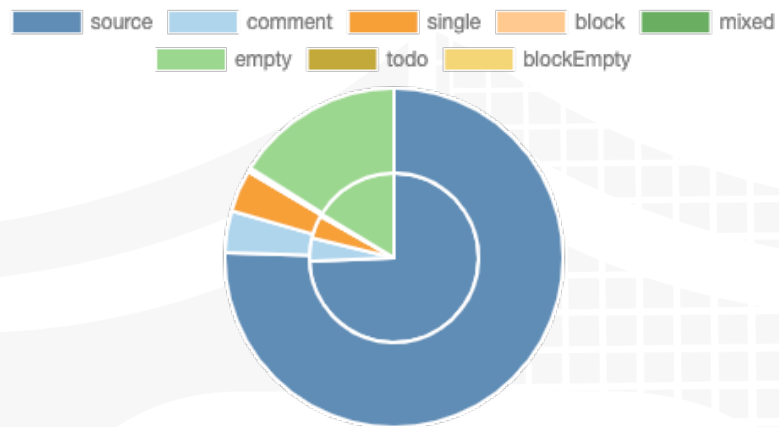
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

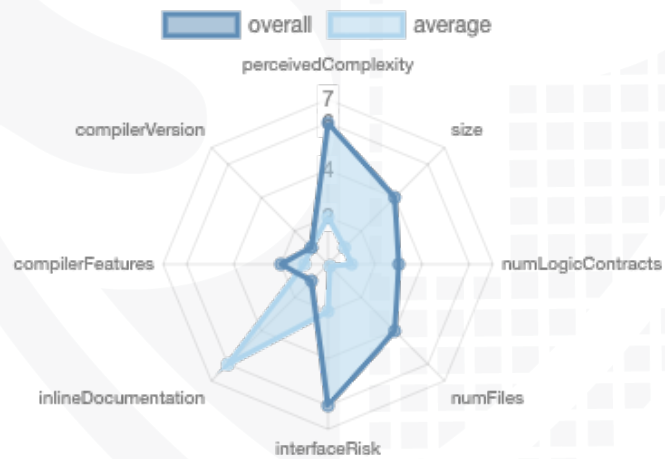
File Name	SHA-1 Hash
contracts/Bank.sol	119f8c95cfa5f2223e1a3095b2b26f983766c953
contracts/Core.sol	0888859f05b26d317c9089d969c1255456c33364
contracts/Voice.sol	c3505175b48d6500501a38cff4c33c37a92f622a
contracts/DevelopmentFund.sol	22b8a55f5d6acc42060e39d56ca1f093055f8b0f
contracts/ExchangeFund.sol	a0b70c7c28e582fd38c35b135ba39e44f55213be
contracts/Baks.sol	c854695fcb334ed778d875c75d3eab84a336109
contracts/BaseToken.sol	79def02ed6512525877953e116c6fc9c47fe4535
contracts/ChainlinkPriceOracle.sol	91277f57ed3cfe0089e5243f1e910c5ada28cd0e
contracts/Depository.sol	58550293d1389c074e056f573a07dc1706ffb66b
contracts/Governance.sol	b71e75981d752afdc3262afe4114e3fada84857a
contracts/libraries/ReentrancyGuard.sol	918902edc9b99b9e5966853f94cc84d6a522aa1c
contracts/interfaces/Chainlink.sol	c247bda0380df398860201f6225360a324f58b7a
contracts/libraries/Deposit.sol	a77d79889a7bd6294d3a8d041a0f645082eff996
contracts/interfaces/IERC165.sol	754045c4544e2677fbff8e619bbb6dd4f4ab0348
contracts/interfaces/EIP173.sol	d23be034f71cb447a9b3eb17ffedaa19013d3cb7
contracts/interfaces/UniswapV2.sol	3db416095fd0b486de1d2565b66aab2b6c42c0f4
contracts/interfaces/EIP2612.sol	762590cf872ae51c556f209730a6b9fda2adf921
contracts/interfaces/IDepository.sol	1cb698001233d270b0490ae4fb2437546aa6fa40
contracts/interfaces/IPriceOracle.sol	cfe142174736a6e7a604c186920da9d916c9be8a
contracts/interfaces/IWrappedNativeCurrency.sol	c71ea61b95dcf0214b504694e9029cf33ea4775d
contracts/interfaces/ERC20.sol	0aa9e66531b96d6245394c4925f02e33664001ce
contracts/interfaces/IBank.sol	74d5f67ea83b9cf63b4e233a5e2eb4e459405bb7
contracts/libraries/Address.sol	e0e6c3de29932c0810e3f502c77ad1b74edb8fed
contracts/libraries/Math.sol	1429c90bdcf252a89a6d4711b6f8d64f850473cc
contracts/libraries/AmountNormalization.sol	854bc4cf8e570c17a653e70aa0474c83a0898efc
contracts/libraries/Magister.sol	c92f545770adc6424e0ae1bd24c9fe236f523d57
contracts/libraries/SafeERC20.sol	4be13e0ddde00aa02bd56151e98d553c32ee870b
contracts/libraries/FixedPointMath.sol	18b008a953b045adb2a6eee6e7c778a3dba12385
contracts/libraries/Upgradability.sol	a10acb5cc5fba62ff2d7e4f9b6856ee16017553d
contracts/libraries/Pool.sol	ca2565ecdf7626679b15a3c51f7b61443cbd62f
contracts/libraries/EnumerableAddressSet.sol	bff81588fe35273f05bb7998a3a9b2adfb6d8a5b
contracts/libraries/CollateralToken.sol	ce741e9a579a477d37ec282206007a575cddc99f
contracts/libraries/Loan.sol	be551d43ee22b3093f35fef2c41fa0b8bc400c53

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	10	12	14	5

Exposed Functions

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

Version	Public	Payable
1.0	153	6

Version	External	Internal	Private	Pure	View
1.0	142	166	0	19	86

State Variables

Version	Total	Public
1.0	85	53

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.10		yes	yes (13 asm blocks)	

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

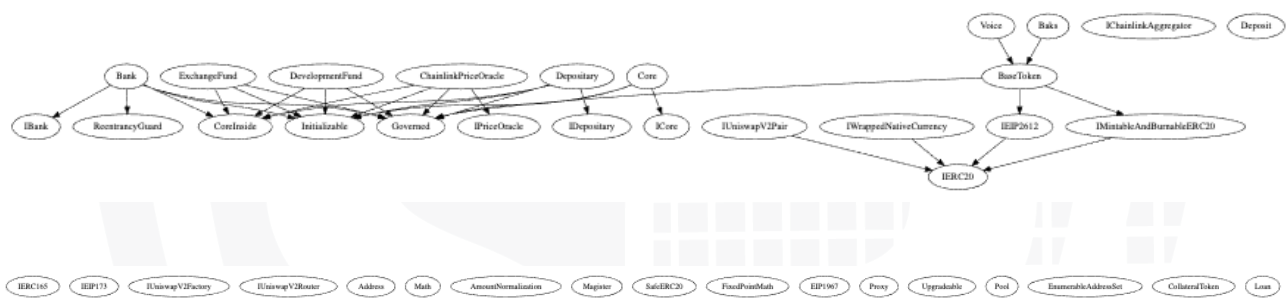
Scope of Work

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph v1.0



Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract

BAKS	BANK	CHAINLINKPRICEORACLE
acceptGovernance	acceptGovernance	acceptGovernance
approve	borrow	initialize
burn	borrowInNativeCurrency	setAggregator
mint	deposit	transitGovernance
permit	depositInNativeCurrency	
transfer	initialize	
setMinter	liquidate	
transferFrom	listCollateralToken	
transitGovernance	onNewDeposit	
	rebalance	
	repay	
	salvage	
	setInitialLoanToValueRatio	
	transitGovernance	
	unlistCollateralToken	

CORE	DEPOSITARY	EXCHANGEFUND	VOICE
acceptGovernance		acceptGovernance	acceptGovernance
initialize	acceptGovernance	deposit	approve
setBaks		divest	burn
setBank	addPool	initialize	mint
setDepositary	blacklistMagister	invest	permit
setDevelopmentFund		listDepositableToken	transfer
setEarlyWithdrawalFee	deposit	salvage	setMinter
setEarlyWithdrawalPeriod	deposit	service	transferFrom
setExchangeFund	initialize	setSlippageTolerance	transitGovernance
setInterest		setSwapDeadline	
setLiquidationLoanToValueRatio	transitGovernance	swap	
setLiquidator		transitGovernance	
setMarginCallLoanToValueRatio	updatePool	unlistDepositableToken	
setMinimumLiquidity	whitelistMagister	withdraw	
setMinimumMagisterDepositA...	withdraw		
setMinimumPrincipalAmount			
setOperator			
setPlatformFees	DEVELOPMENTFUND		
setPriceOracle	acceptGovernance		
setRebalancingThreshold			
setServicingThreshold	initialize		
setStabilityFee			
setVoice	transitGovernance		
setWorkFee			
transitGovernance			

Deployer cannot mint any new tokens

File	Name	Exist	Tested	Verified
Baks	cannot mint	✓	✓	✗
Voice	cannot mint	✓	✓	✗

Max / Total Supply:

Comments

v1.0

- onlyMinter can mint Baks & Voice

Deployer cannot burn or lock user funds

File	Name	Exist	Tested	Verified
Baks	cannot lock	✓	✓	✓
	cannot burn	✓	✓	✗
Voice	cannot lock	✓	✓	✓
	cannot burn	✓	✓	✗

Alleviation

- 1) Baks can be minted or burned ONLY based on collateral value locked.
- 2) BDV can be minted only by achieving LTV based on volumes described in White Paper. BDV can be burned only after repurchase by BaksDAO network.

Burn and Mint are required for this contract to work as specified. Deployer cannot pause the contract.

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Comments

The whole code was compiled and deployed on local testnet.
See testing section for more details.

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	—

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
All	cannot pause	—	—	—

Modifiers

Baks/Voice

- onlyMinter
 - Mint
 - Burn
- onlyGovernor
 - setMinter

Bank

◆ initialize	◆ onNewDeposit
Ⓜ initializer	Ⓜ onlyDepository
◆ borrow	◆ listCollateralToken
Ⓜ tokenAllowedAsCollateral	Ⓜ onlyGovernor
◆ borrowInNativeCurrency 💰	◆ unlistCollateralToken
Ⓜ nonReentrant	Ⓜ onlyGovernor
◆ deposit	◆ setInitialLoanToValueRatio
Ⓜ onActiveLoan	Ⓜ onlyGovernor
Ⓜ notOnSubjectToLiquidation	◆ salvage
◆ depositInNativeCurrency 💰	Ⓜ onlyGovernor
Ⓜ nonReentrant	
Ⓜ onActiveLoan	
Ⓜ notOnSubjectToLiquidation	
◆ repay	
Ⓜ nonReentrant	
Ⓜ onActiveLoan	
Ⓜ notOnSubjectToLiquidation	
◆ liquidate	
Ⓜ onActiveLoan	
Ⓜ onSubjectToLiquidation	

ChainlinkPriceOracle

- ◆ initialize
- Ⓜ initializer
- ◆ setAggregator
- Ⓜ onlyGovernor

Core

- ◆ initialize
- Ⓜ initializer

- onlyGovernor

- ◆ setPriceOracle
- ◆ setBaks
- ◆ setVoice
- ◆ setBank
- ◆ setDepositary
- ◆ setExchangeFund
- ◆ setDevelopmentFund
- ◆ setOperator
- ◆ setLiquidator
- ◆ setInterest
- ◆ setMinimumPrincipalAmount
- ◆ setStabilityFee
- ◆ setPlatformFees
- ◆ setMarginCallLoanToValueRatio
- ◆ setLiquidationLoanToValueRatio
- ◆ setRebalancingThreshold
- ◆ setMinimumMagisterDepositAmount
- ◆ setWorkFee
- ◆ setEarlyWithdrawalPeriod
- ◆ setEarlyWithdrawalFee
- ◆ setServicingThreshold
- ◆ setMinimumLiquidity

Depository

- ♦ initialize
- Ⓜ initializer

- ♦ whitelistMagister
- Ⓜ onlyGovernor
- ♦ blacklistMagister
- Ⓜ onlyGovernor
- ♦ addPool
- Ⓜ onlyGovernor
- ♦ updatePool
- Ⓜ onlyGovernor

DevelopmentFund

- Initialize
 - Initializer

ExchangeFund

- Initialize
 - Initializer

- ♦ deposit
- Ⓜ tokenAllowedToBeDeposited
- ♦ swap
- Ⓜ tokenAllowedToBeSwapped




























- ♦ listDepositToken
- Ⓜ onlyGovernor
- ♦ unlistDepositToken
- Ⓜ onlyGovernor
- ♦ setSlippageTolerance
- Ⓜ onlyGovernor
- ♦ setSwapDeadline
- Ⓜ onlyGovernor
- ♦ salvage
- Ⓜ onlyGovernor
- ♦ service

Governance

- transitGovernance
 - onlyGovernor

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Bank.sol	1	————	605	555	400	50	342	
	contracts/Core.sol	2	1	325	263	204	22	187	————
	contracts/Voice.sol	1	————	8	8	5	1	4	————
	contracts/DevelopmentFund.sol	1	————	13	13	10	1	12	————
	contracts/ExchangeFund.sol	1	————	402	397	294	23	319	————
	contracts/Baks.sol	1	————	8	8	5	1	4	————
	contracts/BaseToken.sol	1	————	226	206	162	2	103	
	contracts/ChainlinkPriceOracle.sol	1	————	88	84	66	1	54	————
	contracts/Depository.sol	1	————	422	387	323	2	225	
	contracts/Governance.sol	1	————	52	52	42	1	23	————
	contracts/libraries/ReentrancyGuard.sol	1	————	27	27	18	1	6	————
	contracts/interfaces/Chainlink.sol	————	1	17	5	3	1	5	————
	contracts/libraries/Deposit.sol	1	————	27	27	23	1	1	————
	contracts/interfaces/ERC165.sol	————	1	6	5	3	1	3	
	contracts/interfaces/EIP173.sol	————	1	10	7	4	1	5	————
	contracts/interfaces/UniswapV2.sol	————	3	80	9	6	1	29	————
	contracts/interfaces/EIP2612.sol	————	1	21	7	4	2	9	————
	contracts/interfaces/Depository.sol	————	1	6	5	3	1	3	————
	contracts/interfaces/PriceOracle.sol	————	1	24	23	8	12	3	————
	contracts/interfaces/IWrappedNativeCurrency.sol	————	1	10	7	4	1	10	
	contracts/interfaces/ERC20.sol	————	2	35	9	6	1	26	————
	contracts/interfaces/Bank.sol	————	1	8	7	4	1	3	————
	contracts/libraries/Address.sol	1	————	60	48	39	1	34	
	contracts/libraries/Math.sol	1	————	100	100	90	2	26	————
	contracts/libraries/AmountNormalization.sol	1	————	28	24	19	1	6	————
	contracts/libraries/Magister.sol	1	————	18	18	14	1	1	————
	contracts/libraries/SafeERC20.sol	1	————	39	29	21	1	11	————
	contracts/libraries/FixedPointMath.sol	1	————	374	370	333	6	286	
	contracts/libraries/Upgradability.sol	4	————	113	113	86	1	89	
	contracts/libraries/Pool.sol	1	————	62	58	47	1	15	————
	contracts/libraries/EnumerableAddressSet.sol	1	————	51	51	40	1	10	————
	contracts/libraries/CollateralToken.sol	1	————	129	102	86	1	33	————
	contracts/libraries/Loan.sol	1	————	55	55	45	1	15	————
	Totals	27	14	3449	3079	2417	145	1902	

Legend

Attribute	Description
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, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Core	Missing Zero Address Validation (missing-zero-check)	150, 149, 148, 147, 184, 194, 199, 209, 204, 219, 214, 179, 189	Check that the address is not zero
#3	Baks	Local variables shadowing	7	Rename the local variables that shadow another component

Informational issues

Issue	File	Type	Line	Description
#1	ExchangeFund	Unused return values	74, 73, 77, 76	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic

#2	FixedPo intMath	Functions that are not used	124	Remove unused functions
#3	Math	Functions that are not used	9	Remove unused functions
#4	Bank	Unused state variables	112	Remove unused state variables
#5	Exchan geFund	Unused state variables	23, 22	Remove unused state variables

Audit Comments

13. December 2021:

- [Read whole report for more information](#)

BaksDAO Test Protocol

Because the repository is private, we chose not to run a public test on the testnet and instead used a local Ganache installation for the subsequent tests.

Compiling successful

Compiling 36 files with 0.8.10

Generating typings for: 44 artifacts in dir: ./src/types/ for target: ethers-v5

Successfully generated 65 typings!

Compilation finished successfully

Note: Solidity 0.8.10 is not fully supported by Hardhat yet. Hardhat can still be used, but some features, like stack traces, might not work correctly.

Deployment

All contracts were deployed on local running blockchain node. No excessive gas usages were detected.

Conclusion

The core functions such as deposit and withdraw also work manually on the blockchain as indicated in the unit tests. Corresponding events are fired. Custom claims were not specified, the basic functions that can be constructed from the whitepaper and the unit tests were tested.

The contract is safe to deploy and basic logic errors were not detected. The code is written to the best standard and sufficiently commented. Known risks were checked by the auditor and the code was scanned for other vulnerabilities as well.

All unit tests make sense and have also been checked. Logic errors could not be found here either.

Disclaimer

We have checked and verified the code to the best of our knowledge. However, deeper logic errors cannot be excluded and Solidproof.io cannot be held liable for any damage that may occur.

All Unit Tests successful

42 passing (5s)

Bank

governed functions

- ✓ lists and unlists token as collateral (58ms)

initial loan-to-value ratio

- ✓ reverts when token is not allowed as collateral (44ms)
- ✓ reverts when new initial loan-to-value ratio equal or higher than margin call loan-to-value ratio
- ✓ updates initial loan-to-value ratio

salvation

- ✓ reverts when trying to salvage one of collateral tokens
- ✓ reverts when trying to salvage stablecoin

borrow

- ✓ reverts when token is not allowed as collateral
- ✓ reverts when `amount` is zero (47ms)
- ✓ reverts when user don't approve a sufficient amount of collateral tokens (40ms)
- ✓ mints correct amount of stablecoin and creates a loan (171ms)

loan interactions

deposit

- ✓ reverts on inactive loan
- ✓ reverts when `amount` is zero
- ✓ reverts when user don't approve a sufficient amount of collateral tokens
- ✓ successfully deposits

repay

- ✓ reverts when loan is subject to liquidation
- ✓ reverts when `amount` is zero

liquidation

- ✓ should revert when loan is not subject to liquidation
- ✓ should liquidate loan

Base Token

- ✓ have correct minter address

metadata

- ✓ returns the correct name
- ✓ returns the correct symbol
- ✓ returns the correct number of decimals

total supply

- ✓ returns the total supply of tokens

Core

bank

- ✓ updates stability fee
- ✓ updates margin call loan-to-value ratio
- ✓ updates liquidation loan-to-value ratio

platform fees

- ✓ reverts when platform fees sum up to less than one
- ✓ reverts when platform fees sum up to more than one
- ✓ updates platform fees

stabilization fund

- ✓ updates rebalancing threshold

exchange fund

- ✓ updates minimum liquidity
- ✓ updates servicing threshold

Exchange Fund

governed functions

- ✓ lists and unlists depositable token (47ms)
- ✓ updates slippage tolerance
- ✓ updates swap deadline
- ✓ service (stableCoin=>token) and need addLiquidity (67ms)
- ✓ service (stableCoin=>token) and not need addLiquidity (49ms)
- ✓ service (token=>stableCoin) and need addLiquidity (83ms)
- ✓ service (token=>stableCoin) and not need addLiquidity (68ms)

Math

- ✓ sqrt
- ✓ fpsqrt
- ✓ abs

SWC Attacks

ID	Title	Relationships	Status
SW C-13 6	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-13 5	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-13 4	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-13 3	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-13 2	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-13 1	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SW C-13 0	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-12 9	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-12 8	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-12 7	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-12 5	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-12 4	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-12 3	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-12 2	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-12 1	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-12 0	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-10 9	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-10 8	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-10 7	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-10 6	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-10 5	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-10 4	Unchecked Call Return Value	CWE-252: Unchecked Return Value	NOT PASSED
SW C-10 3	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-10 2	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-10 1	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-10 0	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "SolidProof" in a white, elegant script font. The "P" in "Proof" is significantly larger and more stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY