SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

XSURGE

Audit

Security Assessment

29. March, 2022

For

XSURGE

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	19. March 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	29. March 2022	• Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://xsurge.net/

Telegram
https://t.me/XSURGEDEFI

Twitter
https://twitter.com/XSURGEDEFI

Facebook
https://www.facebook.com/groups/XSURGEDEFI

Instagram
https://www.instagram.com/XSURGEDEFI/

Reddit
https://www.reddit.com/r/XSURGE/

Discord
https://discord.com/invite/XSURGE

Description

Surge is the first of it's kind that only allows for growth. The tokens use very low fees to raise the price floor with every transaction, whether it be buys, sells, or wallet-to-wallet transfers

Project Engagement

During the 19th of March 2022, **XSURGE 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

XSURGE

Contract Link

v1.0

- Provided as files

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 aspossible.
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.

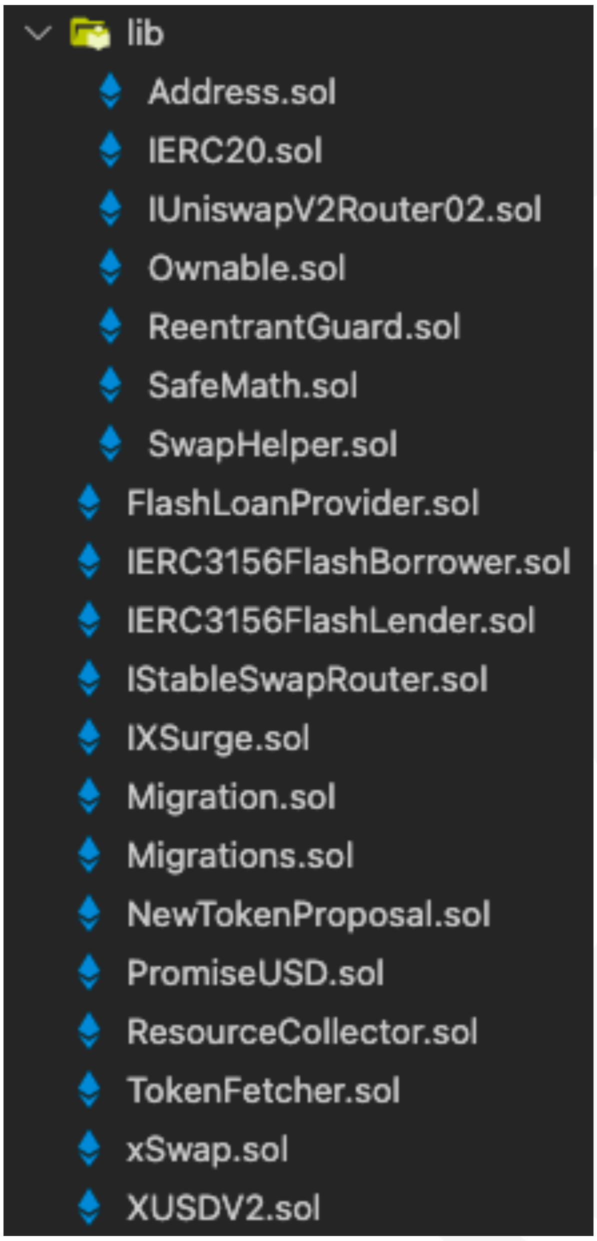
2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to 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:



```
lib
    Address.sol
    IERC20.sol
    IUniswapV2Router02.sol
    Ownable.sol
    ReentrantGuard.sol
    SafeMath.sol
    SwapHelper.sol
  FlashLoanProvider.sol
  IERC3156FlashBorrower.sol
  IERC3156FlashLender.sol
  IStableSwapRouter.sol
  IXSurge.sol
  Migration.sol
  Migrations.sol
  NewTokenProposal.sol
  PromiseUSD.sol
  ResourceCollector.sol
  TokenFetcher.sol
  xSwap.sol
  XUSDV2.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

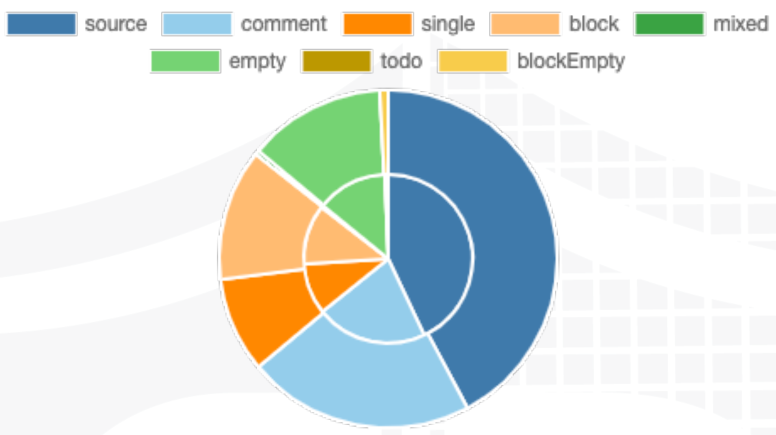
File Name	SHA-1 Hash
contracts/IERC3156FlashLender.sol	6fe140a50b566af15240c67b369eb1f28df2291c
contracts/IStableSwapRouter.sol	ec45d4c4c340c220902aa526bdb3eaa9c1827797
contracts/XUSDV2.sol	54104d577dfb15239237256c97506c0b089f753a
contracts/PromiseUSD.sol	eb1affcc8b9af7c1a239fab272419ee6110e8a4a
contracts/xSwap.sol	7946ffe8dab12dbbb9f5c226452cc2ad4deed09a
contracts/TokenFetcher.sol	39537e173fc21f055ca544875b47f294d532185c
contracts/NewTokenProposal.sol	5a1277c25521223e9f802b03827609844f841a9a
contracts/IXSurge.sol	1f619a8fd54af543e7d6d3c4db952ed7d4713348
contracts/IERC3156FlashBorrower.sol	2731967fc9e337a8bbbd584458d4889f88b58888

v1.1

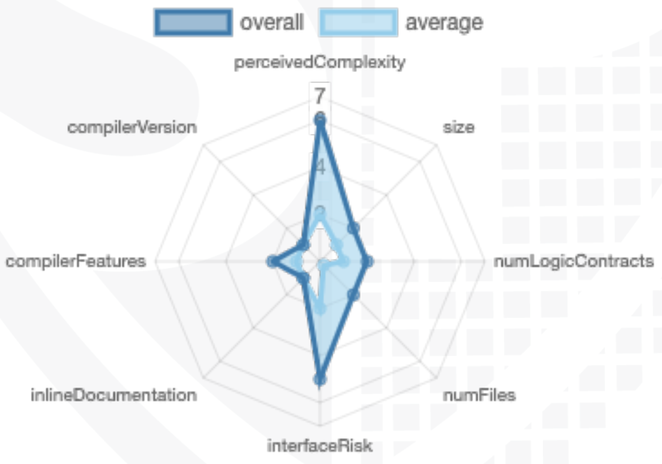
File Name	SHA-1 Hash
contracts/ResourceCollector.sol	baf77cd4de7816efad944073010e04f12e72164b
contracts/IStableSwapRouter.sol	a8c08893362d0cf03e9bef7de0f4e758e12e4b40
contracts/PromiseUSD.sol	70c28d47244fc3cf8b12d0ee5388de95d8815bb9
contracts/xSwap.sol	85c6214c5e7b8aba0ed001d3cbdd4e3eda8c03a7
contracts/IFlashLender.sol	0ea163740b2002aa54089ff4d92795cb7bf10f33
contracts/IFlashBorrower.sol	bf2f942e1efb3ee2bdf0826747a7e3c7a87059ac
contracts/TokenFetcher.sol	232e2e7145c263a6e7b79dc6a1e9c37f00dca075
contracts/NewTokenProposal.sol	940d96838b6df390425e2af503870c31620d93af
contracts/IXSurge.sol	595c0898952695f91168eeb8e97931a506a5a54d
contracts/IERC3156FlashBorrower.sol	2731967fc9e337a8bbbd584458d4889f88b58888
contracts/FlashLoanProvider.sol	7a3ad64b2f97f6d2870c164ee84ceb56a6190c73

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	0	10	0

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	109	5

Version	External	Internal	Private	Pure	View
1.0	92	118	4	7	27

State Variables

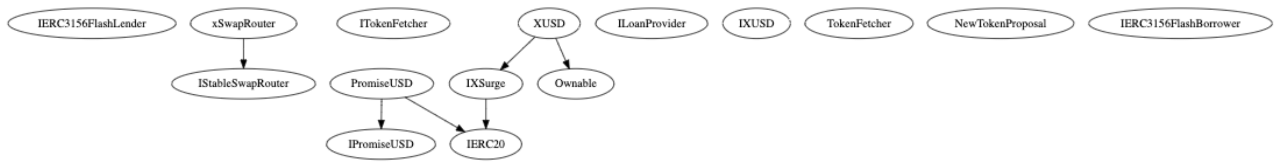
Version	Total	Public
1.0	52	36

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.4`		`yes`		

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

Inheritance Graph
v1.0



CallGraph
v1.0

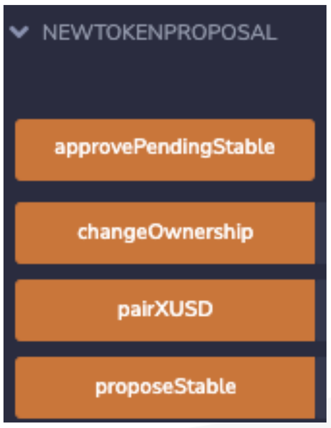
Scope of Work/Verify Claims

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. Overall checkup (Smart Contract Security)

Write functions of contract v1.0

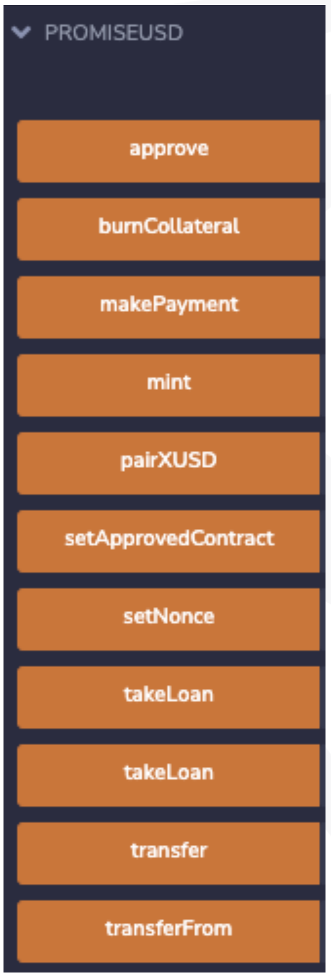

NEWTOKENPROPOSAL

approvePendingStable

changeOwnership

pairXUSD

proposeStable


PROMISEUSD

approve

burnCollateral

makePayment

mint

pairXUSD

setApprovedContract

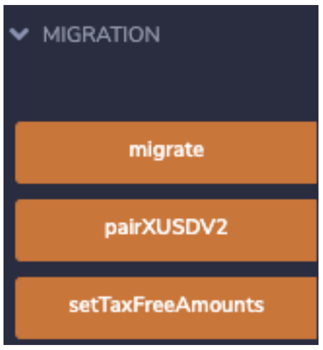
setNonce

takeLoan

takeLoan

transfer

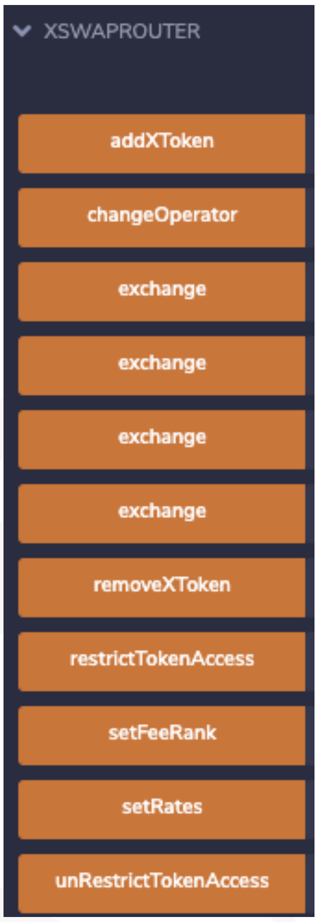
transferFrom


MIGRATION

migrate

pairXUSDV2

setTaxFreeAmounts


XSWAPROUTER

addXToken

changeOperator

exchange

exchange

exchange

exchange

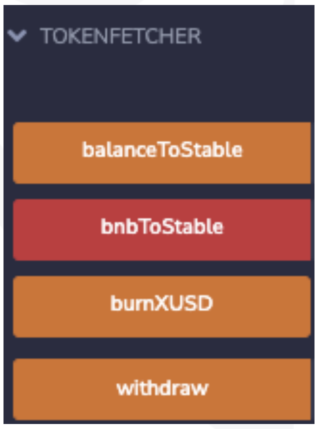
removeXToken

restrictTokenAccess

setFeeRank

setRates

unRestrictTokenAccess

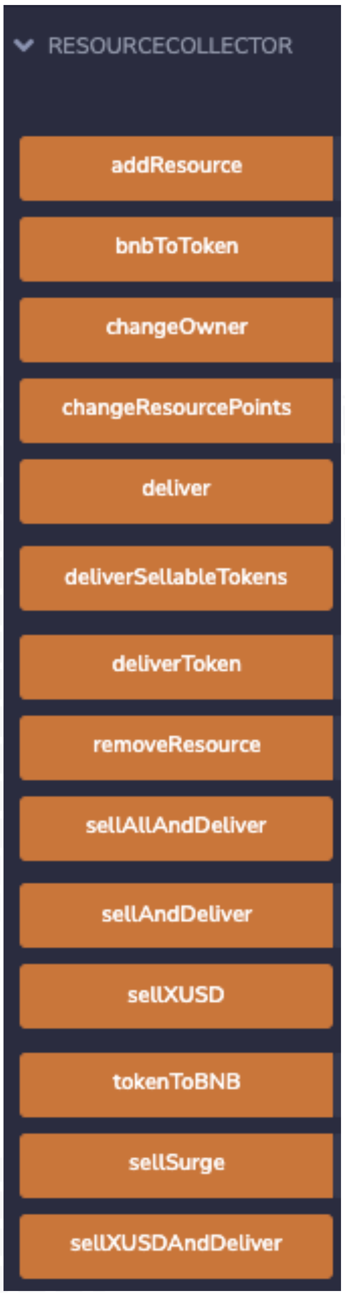

TOKENFETCHER

balanceToStable

bnbToStable

burnXUSD

withdraw


RESOURCECOLLECTOR

addResource

bnbToToken

changeOwner

changeResourcePoints

deliver

deliverSellableTokens

deliverToken

removeResource

sellAllAndDeliver

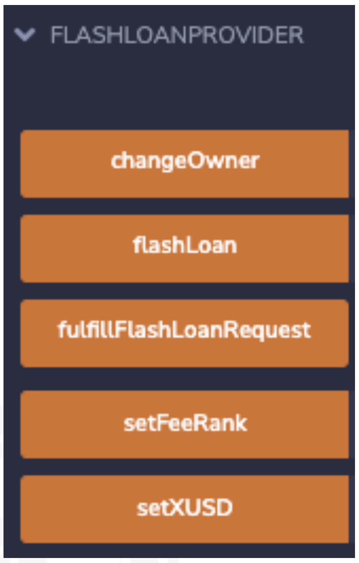
sellAndDeliver

sellXUSD

tokenToBNB

sellSurge

sellXUSDAndDeliver


FLASHLOANPROVIDER

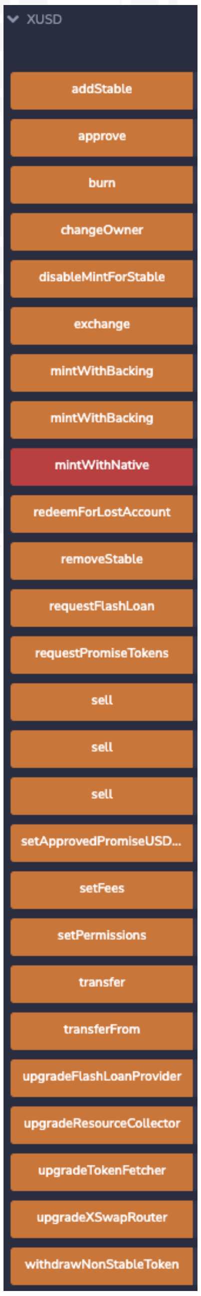
changeOwner

flashLoan

fulfillFlashLoanRequest

setFeeRank

setXUSD


XUSD

addStable

approve

burn

changeOwner

disableMintForStable

exchange

mintWithBacking

mintWithBacking

mintWithNative

redeemForLostAccount

removeStable

requestFlashLoan

requestPromiseTokens

sell

sell

sell

setApprovedPromiseUSD...

setFees

setPermissions

transfer

transferFrom

upgradeFlashLoanProvider

upgradeResourceCollector

upgradeTokenFetcher

upgradeXSwapRouter

withdrawNonStableToken

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

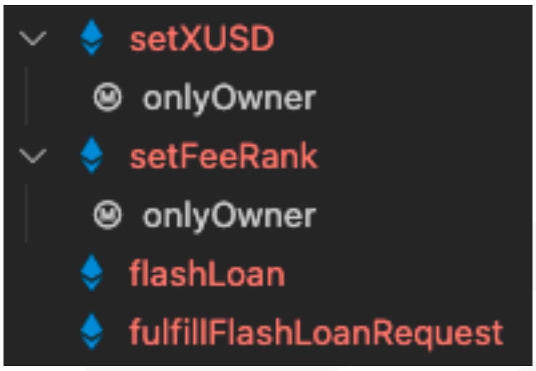
Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

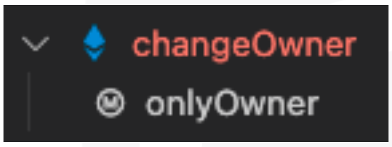
Modifiers and public functions

v1.0

FlashLoanProvider

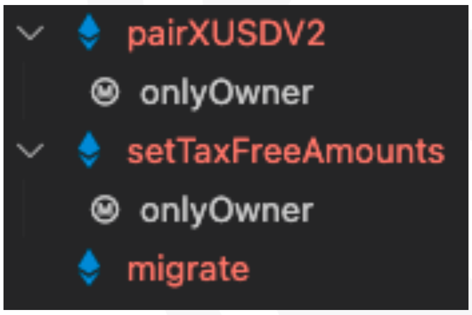


```
setXUSD
    onlyOwner
setFeeRank
    onlyOwner
flashLoan
fulfillFlashLoanRequest
```



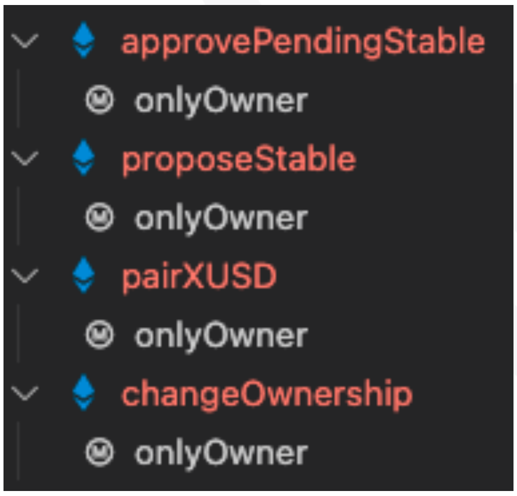
```
changeOwner
    onlyOwner
```

Migration



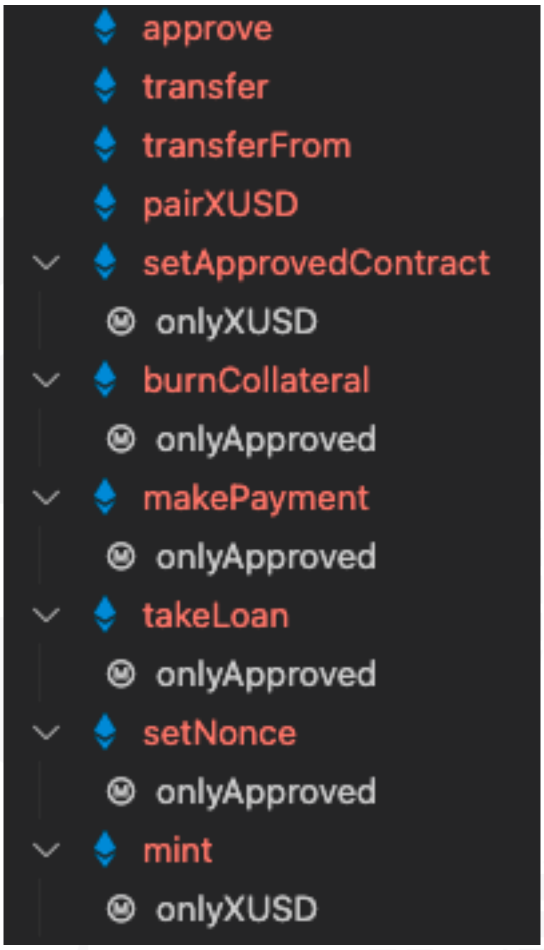
```
pairXUSDV2
    onlyOwner
setTaxFreeAmounts
    onlyOwner
migrate
```

NewTokenProposal



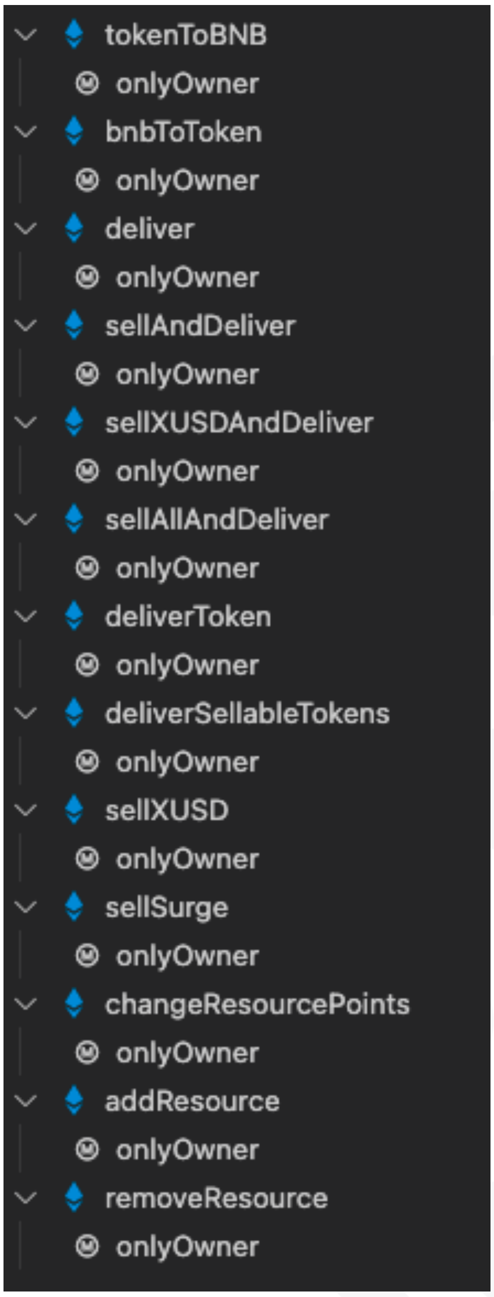
```
approvePendingStable
    onlyOwner
proposeStable
    onlyOwner
pairXUSD
    onlyOwner
changeOwnership
    onlyOwner
```

PromiseUSD



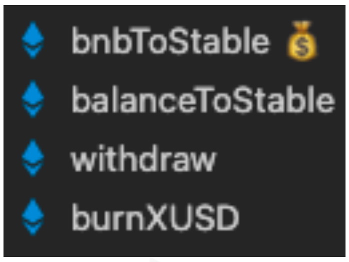
```
approve
transfer
transferFrom
pairXUSD
setApprovedContract
    onlyXUSD
burnCollateral
    onlyApproved
makePayment
    onlyApproved
takeLoan
    onlyApproved
setNonce
    onlyApproved
mint
    onlyXUSD
```

ResourceCollector



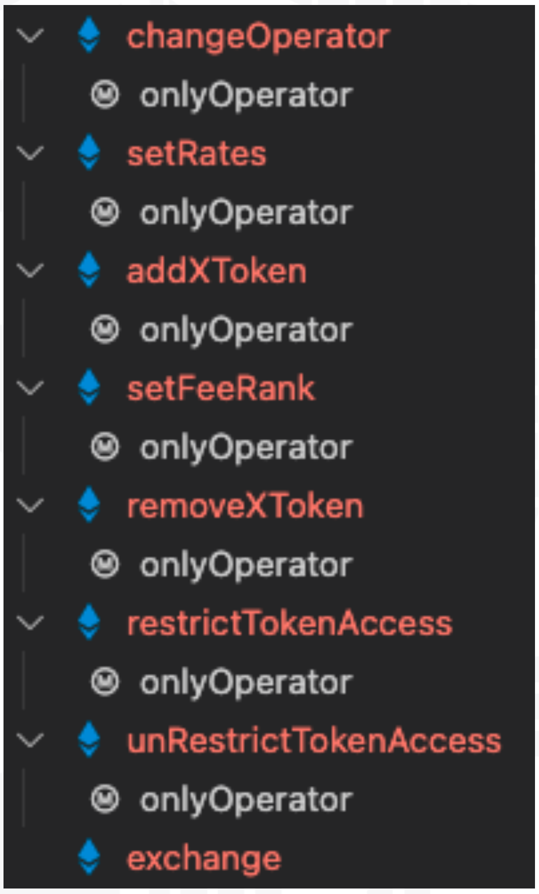
tokenToBNB
onlyOwner
bnbToToken
onlyOwner
deliver
onlyOwner
sellAndDeliver
onlyOwner
sellXUSDAndDeliver
onlyOwner
sellAllAndDeliver
onlyOwner
deliverToken
onlyOwner
deliverSellableTokens
onlyOwner
sellXUSD
onlyOwner
sellSurge
onlyOwner
changeResourcePoints
onlyOwner
addResource
onlyOwner
removeResource
onlyOwner

TokenFetcher



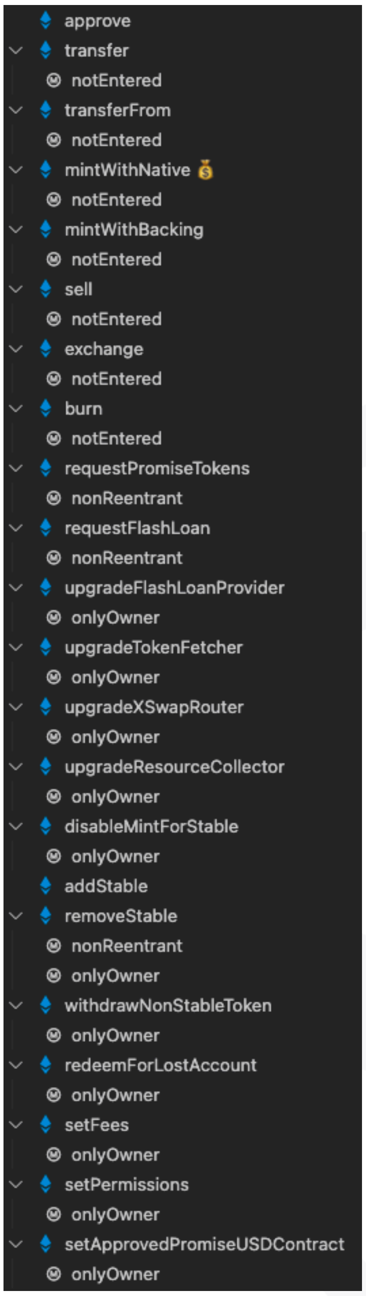
bnbToStable 💰
balanceToStable
withdraw
burnXUSD

xSwap



changeOperator
onlyOperator
setRates
onlyOperator
addXToken
onlyOperator
setFeeRank
onlyOperator
removeXToken
onlyOperator
restrictTokenAccess
onlyOperator
unRestrictTokenAccess
onlyOperator
exchange

XUSDV2



```
approve
transfer
    notEntered
transferFrom
    notEntered
mintWithNative 💰
    notEntered
mintWithBacking
    notEntered
sell
    notEntered
exchange
    notEntered
burn
    notEntered
requestPromiseTokens
    nonReentrant
requestFlashLoan
    nonReentrant
upgradeFlashLoanProvider
    onlyOwner
upgradeTokenFetcher
    onlyOwner
upgradeXSwapRouter
    onlyOwner
upgradeResourceCollector
    onlyOwner
disableMintForStable
    onlyOwner
addStable
removeStable
    nonReentrant
    onlyOwner
withdrawNonStableToken
    onlyOwner
redeemForLostAccount
    onlyOwner
setFees
    onlyOwner
setPermissions
    onlyOwner
setApprovedPromiseUSDContract
    onlyOwner
```

Comments

- **Deployer can set following state variables without any limitations**
    - *Migration.sol*
        - taxFreeAmount
    - *ResourceCollector*
        - receivers[resource].points
    - *XUSDV2*
        - *resourceAllocationPercentage*

- **Deployer can enable/disable following state variables**
    - *xSwap*

        - *tokenDeniedFromSwap[token]*
    - *XUSDV2*
        - stableAssets[stable].mintDisabled
        - isTransferFeeExempt[Contract]

- **Deployer can set following addresses**
    - *FlashLoanProvider.sol*
        - XUSD
            - Only once if address is zero address and the new address isn't
    - *NewTokenProposal*
        - pendingStableToken
        - XUSD
            - Only once if address is zero address and the new address isn't
        - owner
    - *PromiseUSD*
        - XUSD
            - Only once if address is zero address and the new address isn't
        - nonces[msg.sender]
    - *xSwap*
        - operator
        - xTokens[xtoken].resourceCollector
    - *XUSDV2*
        - flashLoanProvider
        - TokenFetcher
        - xSwapRouter
        - resourceCollector

- *FlashLoanProvider*
    - If feeRank is 2 from address, the calculated flash fee will be every time zero in L101
- *Migration*
    - XUSDV can only be paired once
- *PromiseUSD*
    - Only XUSD can mint new tokens
- *ResourceCollector*
    - Owner can send token to bnb
- *XUSDV2*
    - Anybody can
        - Burn

    - Mint
- Fees are set to 0.75% by default but can be set to 2% with setFees function
- Owner can disable minting

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/IERC3156FlashLender.sol	—	1	35	12	4	19	7	🔆
🔍	contracts/IStableSwapRouter.sol	—	1	20	9	3	10	7	—
📝🔍	contracts/XUSDV2.sol	1	3	965	945	518	291	483	💰📤
📝🔍	contracts/PromiseUSD.sol	1	1	381	355	172	149	148	📤
📝	contracts/xSwap.sol	1	—	245	237	169	23	106	📤
📝🔍	contracts/TokenFetcher.sol	1	1	57	54	39	3	50	💰📤
📝🔍	contracts/NewTokenProposal.sol	1	1	73	70	45	10	38	—
🔍	contracts/IXSurge.sol	—	1	21	11	4	5	26	💰
🔍	contracts/IERC3156FlashBorrower.sol	—	1	21	14	3	10	3	🔆
📝🔍	**Totals**	**5**	**10**	**1818**	**1707**	**957**	**520**	**868**	💰📤🔆

v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍	contracts/ResourceCollector.sol	2	2	265	253	171	26	239	📤
🔍	contracts/IStableSwapRouter.sol	—	1	25	9	3	13	9	—
📝🔍	contracts/PromiseUSD.sol	1	1	382	356	173	149	148	📤
📝🔍	contracts/xSwap.sol	1	1	267	238	170	35	125	📤
🔍	contracts/IFlashLender.sol	—	1	37	12	4	20	7	—
🔍	contracts/IFlashBorrower.sol	—	1	23	15	3	11	3	🔆
📝🔍	contracts/TokenFetcher.sol	1	1	89	84	60	6	74	💰📤
📝🔍	contracts/NewTokenProposal.sol	1	2	86	71	46	10	57	—
🔍	contracts/IXSurge.sol	—	1	23	11	4	5	30	💰
🔍	contracts/IERC3156FlashBorrower.sol	—	1	21	14	3	10	3	🔆
📝🔍	contracts/FlashLoanProvider.sol	1	3	251	199	130	53	127	📤🧮🔆
📝🔍	**Totals**	**7**	**15**	**1469**	**1262**	**767**	**338**	**822**	💰📤🧮🔆

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

High issues

No high issues

Medium issues

No medium issues

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	NewTokenProposal	Missing Zero Address Validation (missing-zero-check)	83	Check that the address is not zero
#3	Ownable	Missing Zero Address Validation (missing-zero-check)	39	Check that the address is not zero.
#4	ResourceCollector	Missing Zero Address Validation (missing-zero-check)	118, 122	Check that the address is not zero
#5	XUSDV2	Missing Zero Address Validation (missing-zero-check)	773	Check that the address is not zero

#6	XUSDV2	State variable visibility is not set	44, 47, 48	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	Migration	State variables that could be declared constant (constable-states)	24	Add the `constant` attributes to state variables that never change
#2	Migration	Unused state variables	24	Remove unused state variables
#3	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#4	Reource Collector	Require message missing	184	Provide an error message
#5	XUSDV2	Require message missing	All require statements	Provide an error message

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

19. March 2022:

- Read whole report carefully for more information

29. March 2022:

- Several bugs were fixed by Surge team
- Read whole report carefully for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY