



SOLIDProof
Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

Zyberswap

Stable AMM

Audit

**Security Assessment
13. February, 2023**

For



SolidProof_io



@solidproof_io

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
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	17
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	23
Audit Comments	23
SWC Attacks	24

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 Uniswap, 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. February 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Ethereum (ERC20)

Binance Smart Chain (BEP20)

Website

<https://www.zyberswap.io/>

Telegram

<https://t.me/zyberswap>

Twitter

<https://twitter.com/zyberswap>

Discord

<https://discord.gg/NZ2S3ZEYFj>



Description

Zyberswap is aiming to become one of the first decentralized exchanges (DEX) with an automated market-maker (AMM) on the Arbitrum blockchain. Compared to its competitors, Zyberswap will allow the swapping of crypto assets with **the lowest fees!** Rewards from Staking and Yield Farming will be among **the most lucrative** in the entire Arbitrum ecosystem. Additionally, Zyberswap aims to fully involve its users in decision-making. All major changes will be decided via **Governance Voting!**

Project Engagement

During the 19th of January 2023, **ZyberSwap 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

- Github
 - <https://github.com/Zyberswap-Arbitrum/zyberswap-stableamm-contracts/tree/main/contracts>
 - Commit:
 - <https://github.com/Zyberswap-Arbitrum/zyberswap-stableamm-contracts/commit/cdffaeb901d3496e4b688e1625572c30fad97caa>

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:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/drafts/ERC20PermitUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20BurnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/math/SafeMath.sol	4
@openzeppelin/contracts/proxy/Clones.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	5

Tested Contract Files

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

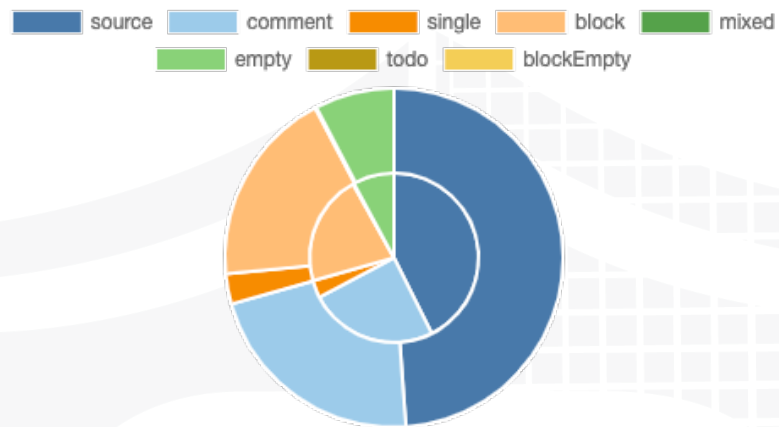
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

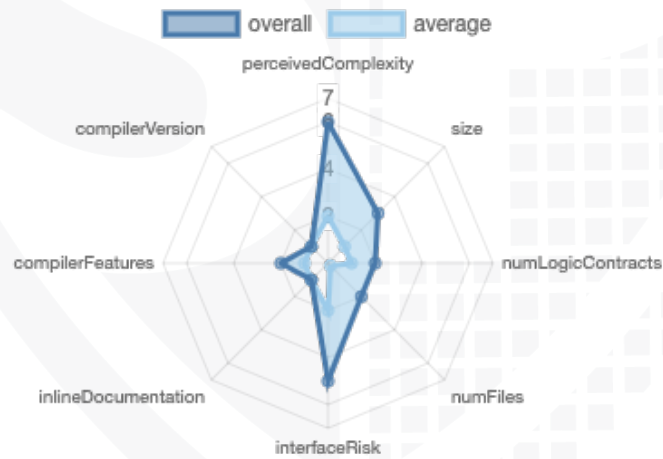
File Name	SHA-1 Hash
contracts/SwapDeployer.sol	5f4a4717edd9112cb03806b344f8738d9b683288
contracts/StableSwapRouter.sol	5191bcfe95b785028171f783027e1e51df9670c0
contracts/LPToken.sol	242077199b8e394d2717bdcd24c8be5a479919ee
contracts/OwnerPausableUpgradeable.sol	c171c6195690985d25cf43c3d6b6833b899903e6
contracts/MathUtils.sol	6a12e4eb0e758c69e0542ce4fa03fdd0cb2f86af
contracts/AmplificationUtils.sol	f1e53909c5aa7c792d778d48ab64ea8880690944
contracts/SwapFlashLoan.sol	a94c6b0e3f3cd0a5e9b84e381eb7d3f1aa6e3f88
contracts/SwapUtils.sol	d7ac1f240b68ddede12811dc82e5c9eeca7a334
contracts/Swap.sol	c86c175350be3c220a7cc83b536c809a7390093f
contracts/interfaces/IFlashLoanReceiver.sol	cdb736405a5a4a0e8650590750b151b3c7b21bbb
contracts/interfaces/IMetaSwap.sol	edd81a53a4dab38001b5fc49feec4f38cbbaaece
contracts/interfaces/ISwap.sol	7aa234a49af818a66a4df2a576b298f032e229c6
contracts/libraries/SafeMath.sol	3b32230115c2f777c54e7b1f31c63f16141652e3
contracts/libraries/Address.sol	464837c0c243ea22d343c93e319ecb7dce4fe336

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	5	3	1

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	104	0

Version	External	Internal	Private	Pure	View
1.0	98	101	1	22	62

State Variables

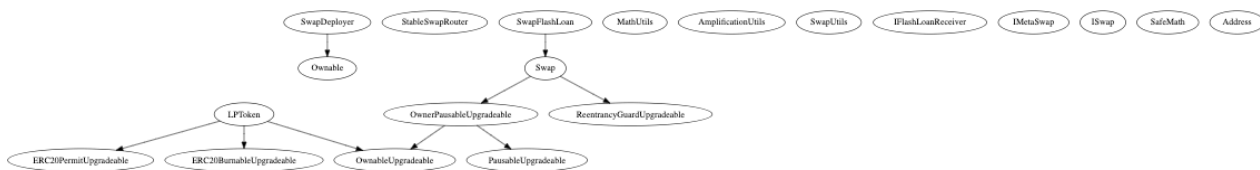
Version	Total	Public
1.0	14	9

Capabilities

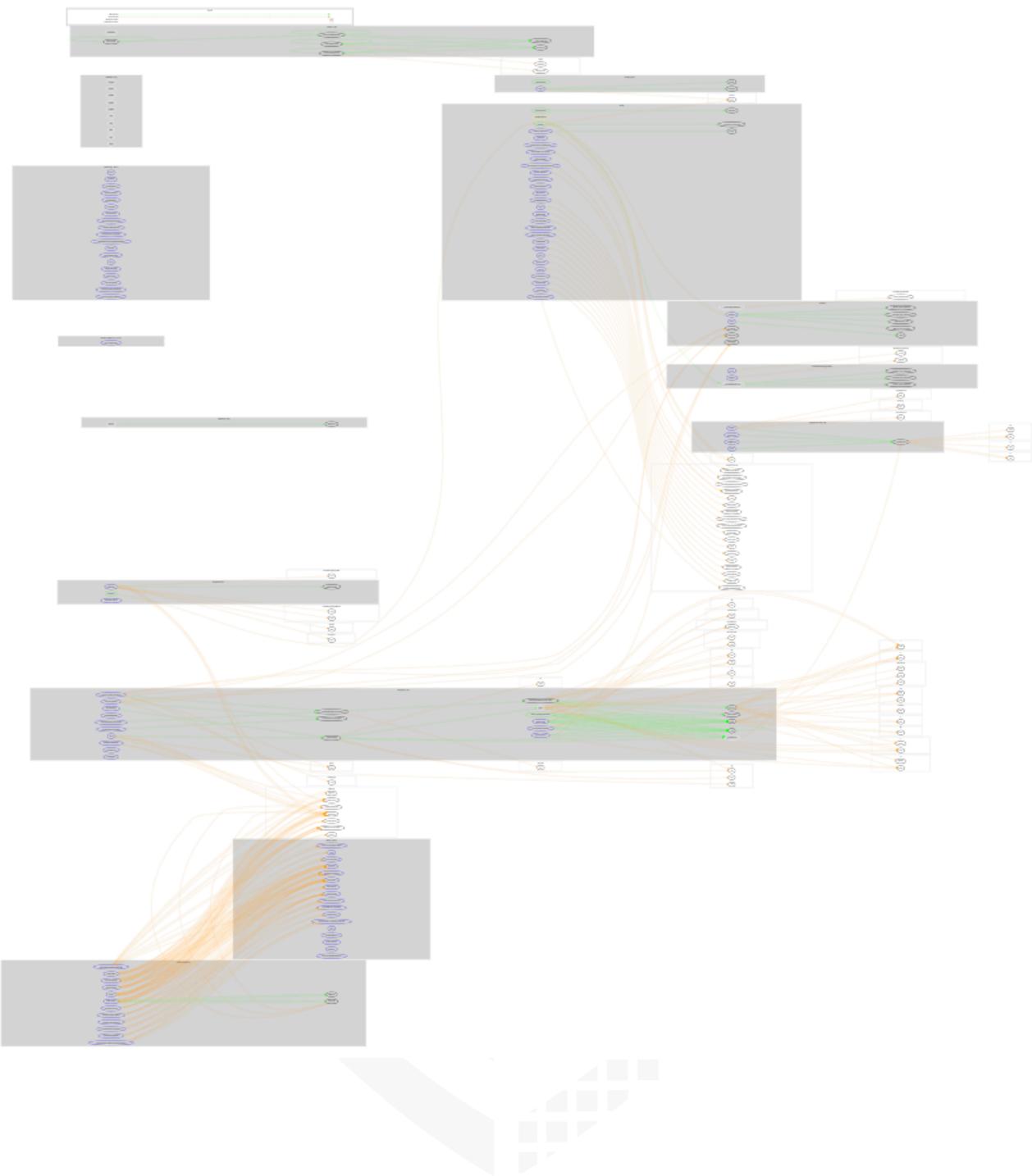
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.6.12 0.7.0 ≥0.6.0 <0.8.0 ≥0.6.2 <0.8.0			yes (2 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	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)



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers and public functions

v1.0

LPToken

```
└─ initialize
    └─ initializer
└─ mint
    └─ onlyOwner
```

OwnerPausableUpgradeable

```
└─ pause
    └─ onlyOwner
└─ unpause
    └─ onlyOwner
```

StableSwapRouter

```
└─ convert
└─ addLiquidity
└─ removeLiquidity
└─ removeBaseLiquidityOneToken
└─ swapFromBase
└─ swapToBase
```

SwapDeployer

```
└─ deploy
```

SwapFlashLoan

```
└─ initialize
    └─ initializer
└─ flashLoan
    └─ nonReentrant
└─ setFlashLoanFees
    └─ onlyOwner
```

Swap

```
└─ initialize
    └─ initializer
└─ swap
    └─ nonReentrant
    └─ whenNotPaused
    └─ deadlineCheck
└─ addLiquidity
    └─ nonReentrant
    └─ whenNotPaused
    └─ deadlineCheck
└─ removeLiquidity
    └─ nonReentrant
    └─ deadlineCheck
└─ removeLiquidityOneToken
    └─ nonReentrant
    └─ whenNotPaused
    └─ deadlineCheck
└─ removeLiquidityImbalance
    └─ nonReentrant
    └─ whenNotPaused
    └─ deadlineCheck
└─ withdrawAdminFees
    └─ onlyOwner
└─ setAdminFee
    └─ onlyOwner
└─ setSwapFee
    └─ onlyOwner
└─ rampA
    └─ onlyOwner
└─ stopRampA
    └─ onlyOwner
```

Note:

- Imported functions from libraries e.g. OpenZeppelin were not listed here
- General fork from saddle finance
- Differences:
 - SwapUtils.sol
 - event WithdrawAdminFee added L60
 - Accidentally wrote a 3 in the comment L387
 - We recommend you to remove it
 - return first depositer if totalsupply is 0
 - emitting withdrawAdminFee L1038
 - SwapFlashLoan.sol
 - added safemath and safeERC20 to the contract
 - removed
 - payable from initialize function L76
 - payable from flashloan function L109
 - payable from setFlashLoanFees function L159
 - SwapDeployer.sol
 - Removed
 - Clone function
 - _clone function
 - deployMetaSwap function
 - Swap.sol
 - Removed
 - NewWithdrawFee event L83
 - payable from function
 - initialize L118
 - swap L398
 - addLiquidity L394
 - removeLiquidityOneToken L445
 - removeLiquidityImbalance L476
 - withdrawAdminFees L491
 - setAdminFee L499
 - setSwapFee L507
 - rampA L518
 - stopRampA L529
 - Added functions
 - getTokens
 - getTokenPrecisionMultipliers
 - getNumberOfTokens
 - getTokenBalances
 - getLpToken
 - getAdminBalances

- [LPToken.sol](#)
 - Added
 - safeMathUpgradeable
 - ERC20PermitUpgradeable
 - __ERC20Permit_init call
- [IMetaSwap.sol](#)
 - Removed metaSwapStorage L21
- [ISwap.sol](#)
 - Changed ERC20 to SafeERC20
 - Removed function
 - getAPrecise
 - owner
 - isGuarded
 - paused
 - swapStorage
 - Added function
 - getNumberOfTokens
 - getLpToken
- [Files that are new](#)
 - StableSwapRouter
 - libraries
 - Address
 - SafeMath
- [No changes](#)
 - AmplificationUtils
 - MathUtils
 - OwnerPausableUpgradeable

Ownership Privileges:

- [LPToken.sol](#)
 - Only the owner of the contract is able to mint new tokens
- [OwnerPauseUpgradeable.sol](#)
 - Owner can pause
- [SwapFlashLoan.sol](#)
 - Owner can set to collect 100% of LP fees.

General information

- Be aware of the initialize functions in upgradeables hence it is not protected by a modifier that only an specific address can call it. Basically if a contract is deployed by a framework like e.g. hardhat

the initialize function will be called automatically except the specified initializer function was changed in the deployment.




















- We recommend you to write unit tests for the contracts and test it. Especially the StableSwapRouter

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/SwapDeployer.sol	1	————	44	34	28	1	15	————
	contracts/StableSwapRouter.sol	1	————	387	302	270	4	315	
	contracts/LPToken.sol	1	————	61	53	24	24	22	————
	contracts/OwnerPausableUpgradeable.sol	1	————	37	37	19	13	17	————
	contracts/MathUtils.sol	1	————	37	37	13	20	3	————
	contracts/AmplificationUtils.sol	1	————	159	147	90	44	46	————
	contracts/SwapFlashLoan.sol	1	————	169	152	73	65	40	————
	contracts/SwapUtils.sol	1	————	1068	970	591	265	485	————
	contracts/Swap.sol	1	————	552	454	212	195	177	————
	contracts/interfaces/IFlashLoanReceiver.sol	————	1	19	12	3	8	3	————
	contracts/interfaces/IMetaSwap.sol	————	1	114	10	5	4	39	————
	contracts/interfaces/ISwap.sol	————	1	88	8	4	4	35	————
	contracts/libraries/SafeMath.sol	1	————	246	214	61	139	16	
	contracts/libraries/Address.sol	1	————	253	185	94	113	47	
	Totals	11	3	3234	2615	1487	899	1260	

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	StableSwapRouter	Missing Zero Address Validation (missing-zero-check)	11,12,49, 50, 116, 116, 155, 156, 188, 189, 223, 224, 262, 263, 278, 279, 297, 298, 314, 315, 332, 333, 355, 356,	Check that the address is not zero
#2	LPToken	Local variables shadowing	26	Rename the local variables that shadow another component
#3	SwapFlashLoan	Missing Events Arithmetic	166, 167	Emit an event for critical parameter changes
#4	StableSwapRouter	calculateSwaoFromBase	343	We recommend you to check whether the "tokenIndexFrom" L334 is below the base pool balance length because otherwise you are out of range with the set dx base amount in L343.

#5	StableSwapRouter	min_amount uint256 array	23	The array will be full of zeroes. The array will not be updated while calling convert function
----	------------------	--------------------------	----	--

Informational issues

Issue	File	Type	Line	Description
#1	StableSwapRouter	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#2	SwapDeployer	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#3	SwapUtils	Zero check validation	214	We recommend you to check the TotalSupply variable to 0, because it is divided by this variable.
#4	SwapUtils	Zero check validation	609, 629, 631	We recommend you to check the d0 variable to 0, because it is divided by this variable.
#5	IMetaSwap	Unused interface	-	Remove or use the interface in the project
#6	Address	Unused library	-	Remove or use the library in the project
#7	SafeMath	Unused library	-	Remove or use the local library in the project

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/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.

13. February 2023:

- This project consists of the following forks
 - Saddle Finance
- Read whole report and modifiers section for more information
- Do your own research here

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	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	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-11 1	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-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	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
Development | Marketing**


MADE IN GERMANY