SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

StellaSwap

Audit

Security Assessment

02. February, 2022

For

StellaSwap

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	02. February 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Moonbeam (Polkadot)

Website

https://stellaswap.com/

Telegram

https://t.me/stellaswap

Twitter

https://twitter.com/StellaSwap

Github

https://github.com/stellaswap

Reddit

https://www.reddit.com/user/stellaswap

Medium

https://stellaswap.medium.com/

Description

All your DeFi needs in one place.
Swap, earn and build on Moonbeam's leading DEX

Project Engagement

During the 28th of January 2022, **StellaSwap 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/stellaswap/core
  - Commit: a20e85bc0bacbad189fc4fd8669e4c870f24e5cd

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.

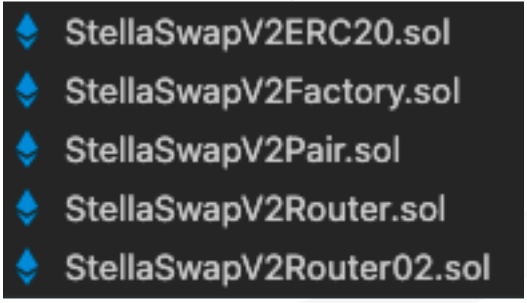
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



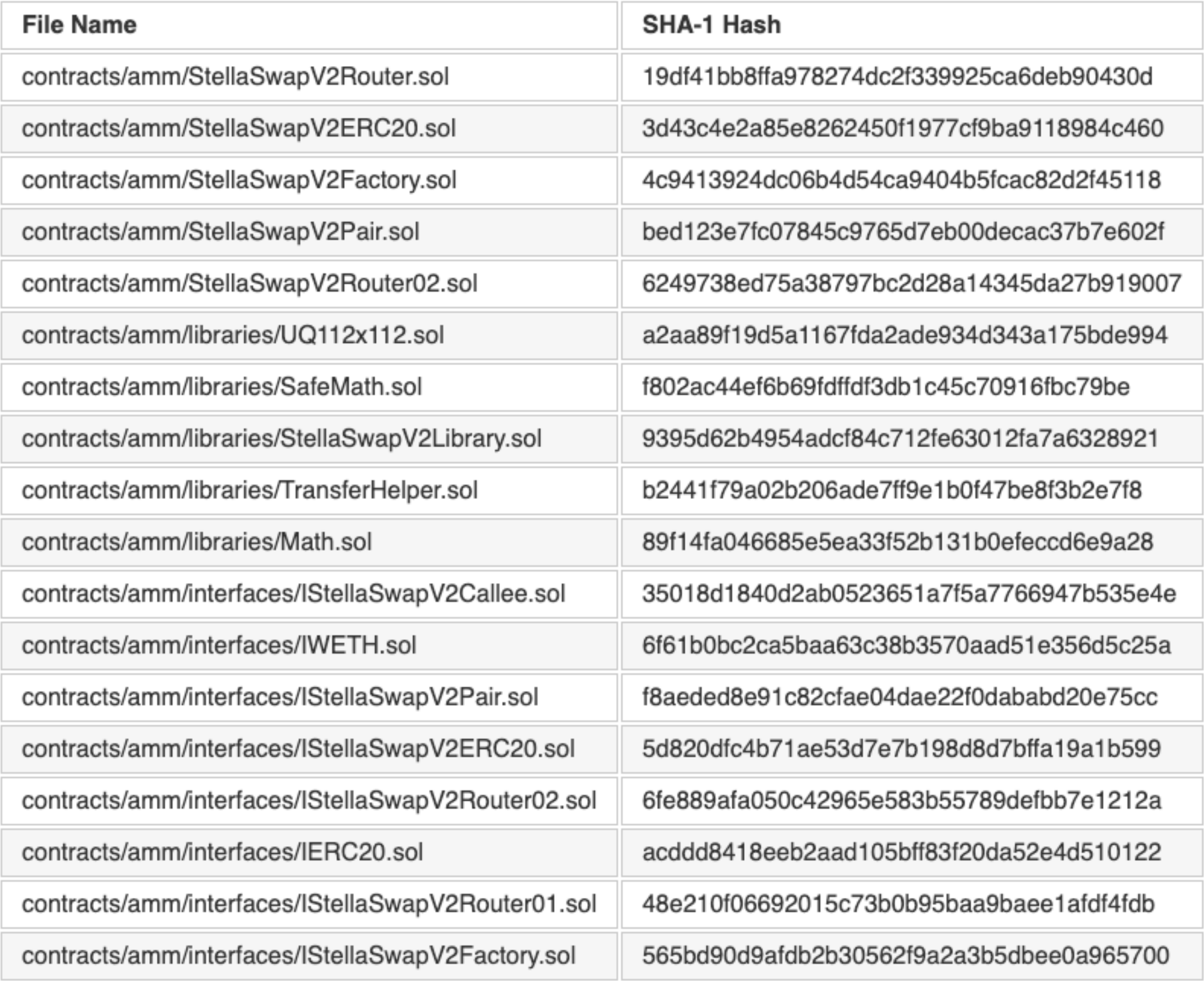
StellaSwapV2ERC20.sol
StellaSwapV2Factory.sol
StellaSwapV2Pair.sol
StellaSwapV2Router.sol
StellaSwapV2Router02.sol

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

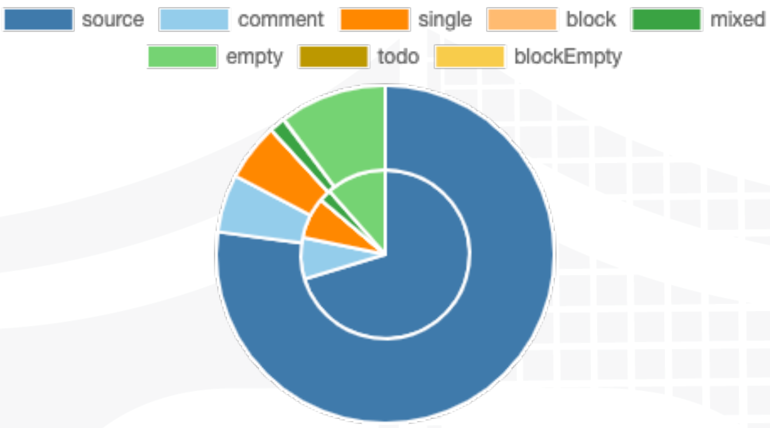


File Name	SHA-1 Hash
contracts/amm/StellaSwapV2Router.sol	19df41bb8ffa978274dc2f339925ca6deb90430d
contracts/amm/StellaSwapV2ERC20.sol	3d43c4e2a85e8262450f1977cf9ba9118984c460
contracts/amm/StellaSwapV2Factory.sol	4c9413924dc06b4d54ca9404b5fcac82d2f45118
contracts/amm/StellaSwapV2Pair.sol	bed123e7fc07845c9765d7eb00decac37b7e602f
contracts/amm/StellaSwapV2Router02.sol	6249738ed75a38797bc2d28a14345da27b919007
contracts/amm/libraries/UQ112x112.sol	a2aa89f19d5a1167fda2ade934d343a175bde994
contracts/amm/libraries/SafeMath.sol	f802ac44ef6b69fdffdf3db1c45c70916fbc79be
contracts/amm/libraries/StellaSwapV2Library.sol	9395d62b4954adcf84c712fe63012fa7a6328921
contracts/amm/libraries/TransferHelper.sol	b2441f79a02b206ade7ff9e1b0f47be8f3b2e7f8
contracts/amm/libraries/Math.sol	89f14fa046685e5ea33f52b131b0efeccd6e9a28
contracts/amm/interfaces/IStellaSwapV2Callee.sol	35018d1840d2ab0523651a7f5a7766947b535e4e
contracts/amm/interfaces/IWETH.sol	6f61b0bc2ca5baa63c38b3570aad51e356d5c25a
contracts/amm/interfaces/IStellaSwapV2Pair.sol	f8aeded8e91c82cfae04dae22f0dababd20e75cc
contracts/amm/interfaces/IStellaSwapV2ERC20.sol	5d820dfc4b71ae53d7e7b198d8d7bffa19a1b599
contracts/amm/interfaces/IStellaSwapV2Router02.sol	6fe889afa050c42965e583b55789defbb7e1212a
contracts/amm/interfaces/IERC20.sol	acddd8418eeb2aad105bff83f20da52e4d510122
contracts/amm/interfaces/IStellaSwapV2Router01.sol	48e210f06692015c73b0b95baa9baee1afdf4fdb
contracts/amm/interfaces/IStellaSwapV2Factory.sol	565bd90d9afdb2b30562f9a2a3b5dbee0a965700

Metrics

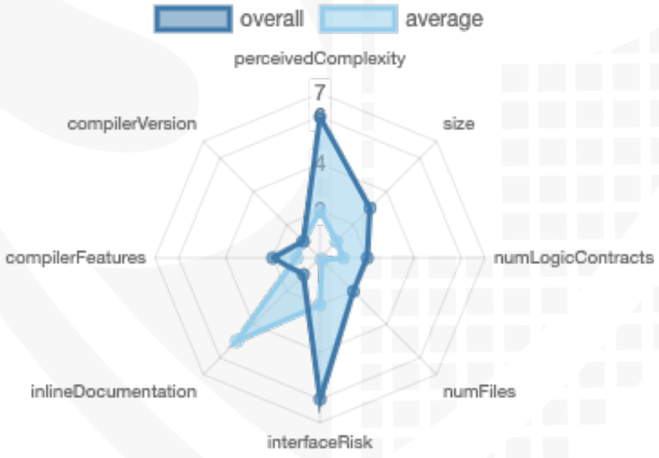
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	8	15	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	233	20

Version	External	Internal	Private	Pure	View
1.0	211	181	5	51	70

State Variables

Version	Total	Public
1.0	34	28

Capabilities

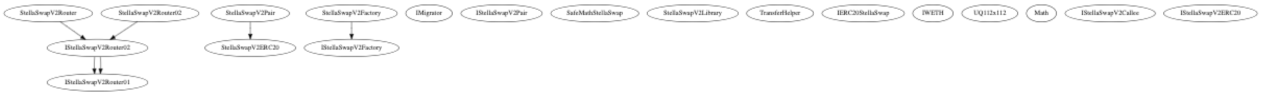
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

1.0	`=0.6.12`
	`>=0.5.0`
	`>=0.6.0`
	`>=0.6.2`
			`yes`	`yes`
				(2 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	`yes`			`yes`	`yes`	`yes` → `AssemblyCall:Name:create2`

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).

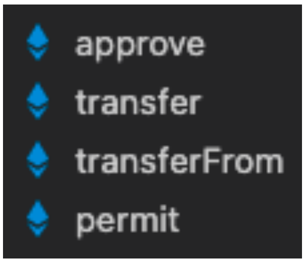
We will verify the following claims:

1. 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)

Correct implementation of Token standard

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 v1.0



approve

transfer

transferFrom

permit

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗
Max / Total Supply	-		

Comments:

v1.0

- Everybody can mint tokens
    - There is no modifier to restrict the mint function

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

- Everybody can burn tokens and transfer token0/token1 to specified address
    - There is no modifier to restrict the burn function

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

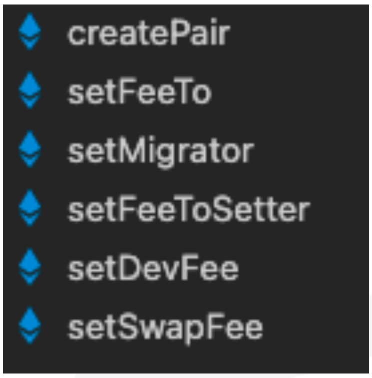
Legend

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

Modifiers and public functions

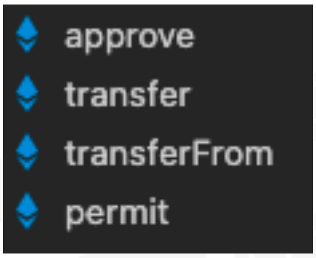
v1.0

StellaSwapV2Factory



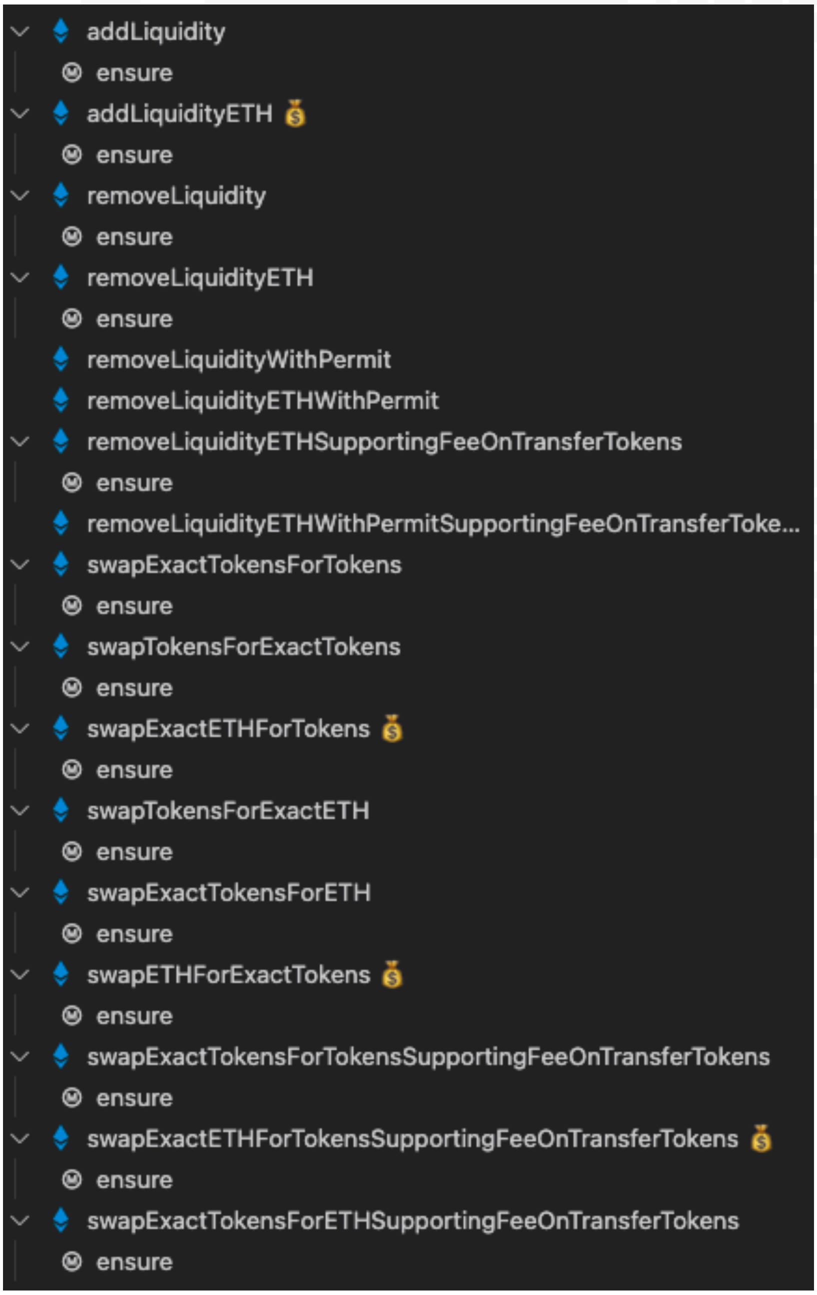
```
createPair
setFeeTo
setMigrator
setFeeToSetter
setDevFee
setSwapFee
```

StellaSwapV2ERC20



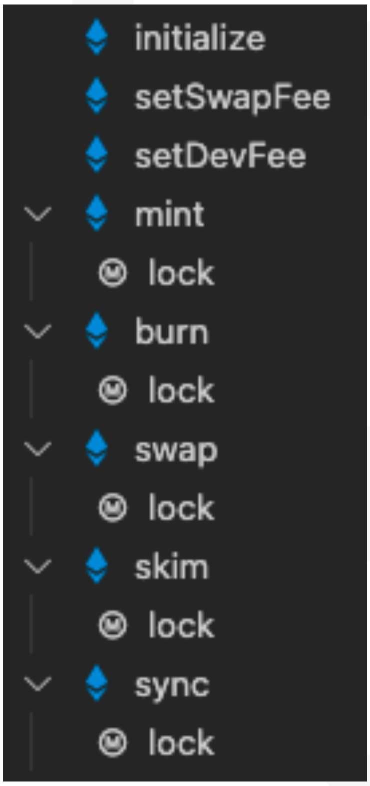
```
approve
transfer
transferFrom
permit
```

StellaSwapV2Router



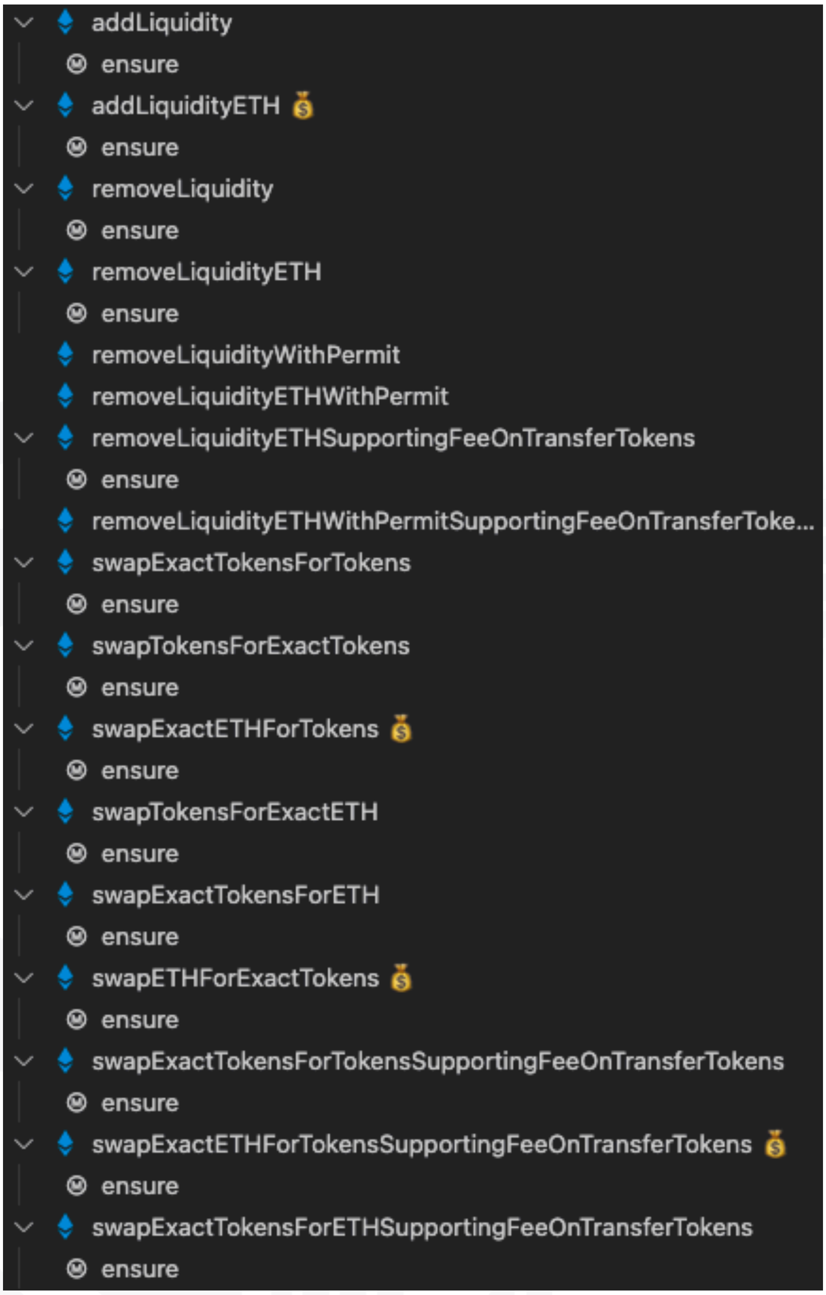
```
addLiquidity
  ensure
addLiquidityETH 💰
  ensure
removeLiquidity
  ensure
removeLiquidityETH
  ensure
removeLiquidityWithPermit
removeLiquidityETHWithPermit
removeLiquidityETHSupportingFeeOnTransferTokens
  ensure
removeLiquidityETHWithPermitSupportingFeeOnTransferToke...
swapExactTokensForTokens
  ensure
swapTokensForExactTokens
  ensure
swapExactETHForTokens 💰
  ensure
swapTokensForExactETH
  ensure
swapExactTokensForETH
  ensure
swapETHForExactTokens 💰
  ensure
swapExactTokensForTokensSupportingFeeOnTransferTokens
  ensure
swapExactETHForTokensSupportingFeeOnTransferTokens 💰
  ensure
swapExactTokensForETHSupportingFeeOnTransferTokens
  ensure
```

StellaSwapV2Pair



```
    initialize
    setSwapFee
    setDevFee
⌄   mint
      lock
⌄   burn
      lock
⌄   swap
      lock
⌄   skim
      lock
⌄   sync
      lock
```

StellaSwapV2Router02



```
⌄ addLiquidity
    ensure
⌄ addLiquidityETH 💰
    ensure
⌄ removeLiquidity
    ensure
⌄ removeLiquidityETH
    ensure
  removeLiquidityWithPermit
  removeLiquidityETHWithPermit
⌄ removeLiquidityETHSupportingFeeOnTransferTokens
    ensure
  removeLiquidityETHWithPermitSupportingFeeOnTransferToke...
⌄ swapExactTokensForTokens
    ensure
⌄ swapTokensForExactTokens
    ensure
⌄ swapExactETHForTokens 💰
    ensure
⌄ swapTokensForExactETH
    ensure
⌄ swapExactTokensForETH
    ensure
⌄ swapETHForExactTokens 💰
    ensure
⌄ swapExactTokensForTokensSupportingFeeOnTransferTokens
    ensure
⌄ swapExactETHForTokensSupportingFeeOnTransferTokens 💰
    ensure
⌄ swapExactTokensForETHSupportingFeeOnTransferTokens
    ensure
```

Comments

- Deployer can set following state variables without any limitations
  - StellaSwapV2Factory
    - feeTo

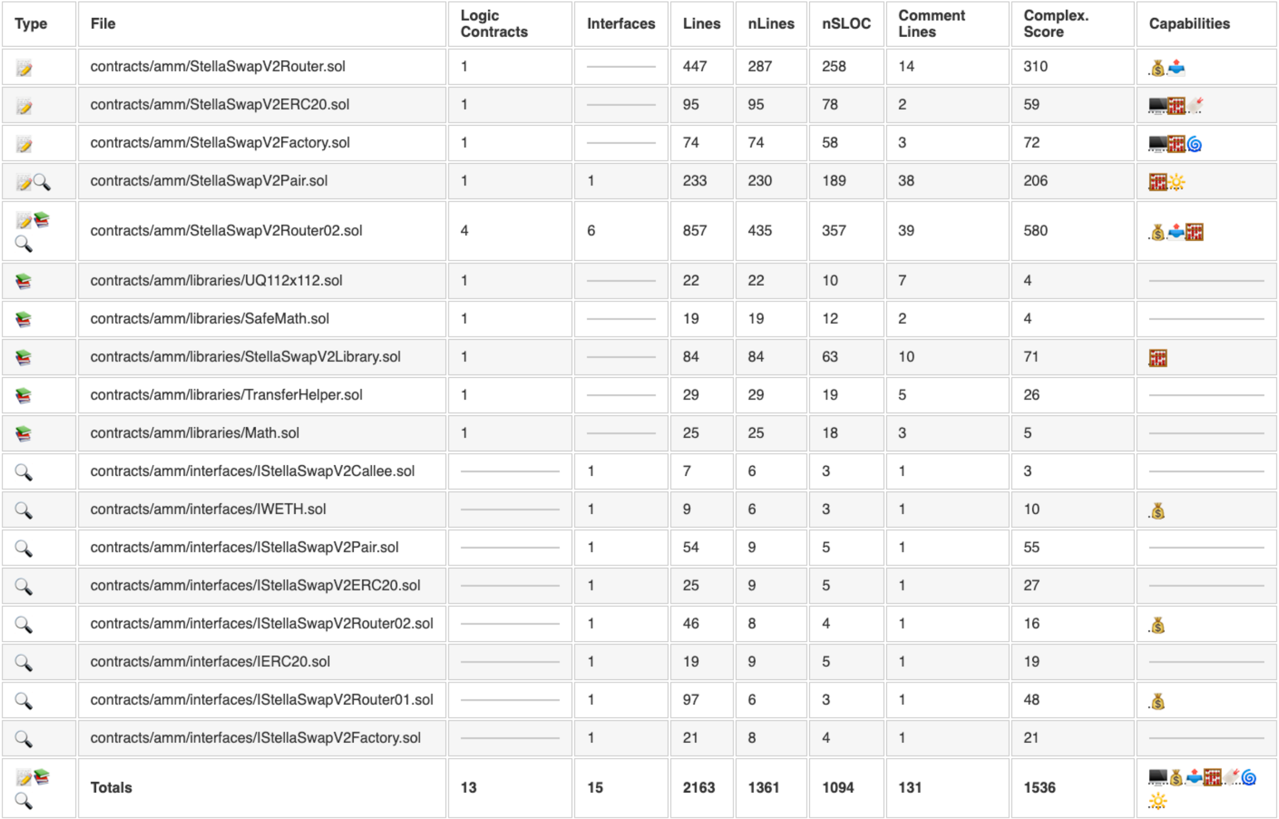
- Everyone can call following functions
  - StellaSwapV2Factory
    - createPair
  - StellaSwapV2Pair
    - mint
    - burn
    - Swap
  - StellaSwapV2Router

- Every external/public function can be called from everyone

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/amm/StellaSwapV2Router.sol	1	—	447	287	258	14	310	💰📤
📝	contracts/amm/StellaSwapV2ERC20.sol	1	—	95	95	78	2	59	🖥🧮🔖
📝	contracts/amm/StellaSwapV2Factory.sol	1	—	74	74	58	3	72	🖥🧮🌀
📝🔍	contracts/amm/StellaSwapV2Pair.sol	1	1	233	230	189	38	206	🧮🔆
📝📚🔍	contracts/amm/StellaSwapV2Router02.sol	4	6	857	435	357	39	580	💰📤🧮
📚	contracts/amm/libraries/UQ112x112.sol	1	—	22	22	10	7	4	—
📚	contracts/amm/libraries/SafeMath.sol	1	—	19	19	12	2	4	—
📚	contracts/amm/libraries/StellaSwapV2Library.sol	1	—	84	84	63	10	71	🧮
📚	contracts/amm/libraries/TransferHelper.sol	1	—	29	29	19	5	26	—
📚	contracts/amm/libraries/Math.sol	1	—	25	25	18	3	5	—
🔍	contracts/amm/interfaces/IStellaSwapV2Callee.sol	—	1	7	6	3	1	3	—
🔍	contracts/amm/interfaces/IWETH.sol	—	1	9	6	3	1	10	💰
🔍	contracts/amm/interfaces/IStellaSwapV2Pair.sol	—	1	54	9	5	1	55	—
🔍	contracts/amm/interfaces/IStellaSwapV2ERC20.sol	—	1	25	9	5	1	27	—
🔍	contracts/amm/interfaces/IStellaSwapV2Router02.sol	—	1	46	8	4	1	16	💰
🔍	contracts/amm/interfaces/IERC20.sol	—	1	19	9	5	1	19	—
🔍	contracts/amm/interfaces/IStellaSwapV2Router01.sol	—	1	97	6	3	1	48	💰
🔍	contracts/amm/interfaces/IStellaSwapV2Factory.sol	—	1	21	8	4	1	21	—
📝📚🔍	**Totals**	**13**	**15**	**2163**	**1361**	**1094**	**131**	**1536**	🖥💰📤🧮🔖🌀🔆

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	All	A floating pragma is set	Lines next to description	The current pragma Solidity directives: • >=0.5.0 (libraries/StellaSwapV2Library.sol:3) • >=0.5.0 (interfaces/IStellaSwapV2Factory.sol#3) • >=0.5.0 (interfaces/IERC20.sol#3) • >=0.5.0 (interfaces/IStellaSwapV2Callee.sol#3) • >=0.6.2 (interfaces/IStellaSwapV2Router02.sol#3) • >=0.5.0 (interfaces/IStellaSwapV2Pair.sol#3) • >=0.6.2 (interfaces/IStellaSwapV2Router01.sol#3) • >=0.6.0 (libraries/TransferHelper.sol#3) • >=0.5.0 (interfaces/IWETH.sol#3) • >=0.5.0 (interfaces/IStellaSwapV2ERC20.sol#3) • >=0.5.0 (StellaSwapV2Router02.sol#7) • >=0.5.0 (StellaSwapV2Router02.sol#88) • >=0.6.0 (StellaSwapV2Router02.sol#172) • >=0.6.2 (StellaSwapV2Router02.sol#205) • >=0.6.2 (StellaSwapV2Router02.sol#306) • >=0.5.0 (StellaSwapV2Router02.sol#354) • >=0.5.0 (StellaSwapV2Router02.sol#379) • >=0.5.0 (StellaSwapV2Router02.sol#402)

#2	StellaSwapV2Factory	Missing Zero Address Validation (missing-zero-check)	19, 48, 58, 53,	Check that the address is not zero
#3	StellaSwapV2Pair	Missing Zero Address Validation (missing-zero-check)	75	Check that the address is not zero
#4	StellaSwapV2Router	Missing Zero Address Validation (missing-zero-check)	24	Check that the address is not zero
#5	StellaSwapV2Router02	Missing Zero Address Validation (missing-zero-check)	434	Check that the address is not zero
#6	StellaSwapV2Pair	Missing Events Arithmetic	88, 81	Emit an event for critical parameter changes

Informational issues

No informational issues

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.

02. February 2022:

- Read whole report 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	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	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

´

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY