SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

StellaSwap

Audit

Security Assessment

28.July,2022

For

SolidProof_io

@solidproof_io

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	26.July,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Binance (BSC)

Website

https://stellaswap.com/

Twitter

https://twitter.com/StellaSwap

Telegram

https://t.me/stellaswap

Github

https://github.com/stellaswap

Discord

https://discord.stellaswap.com/

Reddit

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

Description

StellaSwap is one of the first automated market-making (AMM), decentralized exchange (DEX) for the Moonbeam parachain network. The unique value proposition of StellaSwap is that we're committed in establishing a strong foundation with our native token, STELLA, as a governance token, diverse farms, a built in bridge and user-centered service.
StellaSwap's main objective is to create a broader range of network effects to address the issues of liquidity in the DeFi space, instead of limiting ourselves to a single solution like many DEXs are doing now. This manifests itself in the diverse product suite of StellaSwap that will be explained in more details. Our products are structured in such a way that facilitates decentralized governance of STELLA holders, while continuing to innovate on the collective foundations by design.

Project Engagement

During the 26th of July 2022, **StellaSwap** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Links

v1.0

https://github.com/stellaswap/stellas-ido

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 analyzing 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:

```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
@openzeppelin/contracts/utils/structs/EnumerableSet.sol
./TransferHelper.sol
> { } pragma ... (1)
> IStellaERC20
> IIDOLocker
> IDOSale                                  +9
```

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.

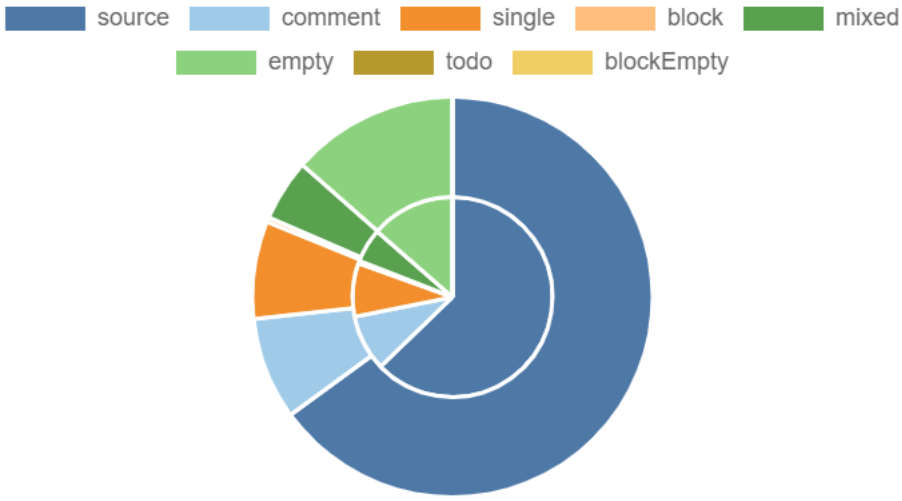
v1.0

File Name	SHA-1 Hash
contracts/ido/TransferHelper.sol	b2441f79a02b206ade7ff9e1b0f47be8f3b2e7f8
contracts/ido/IDOLocker.sol	67bebfd8be52e0b29e4c6919b5357c9c26867f6d
contracts/ido/IDOSale.sol	f253094f132a474b143f8994d635640cdb734320
contracts/utils/MockERC20.sol	602796122bfc5de88541922ee0b22871b1e4fb2f

Metrics

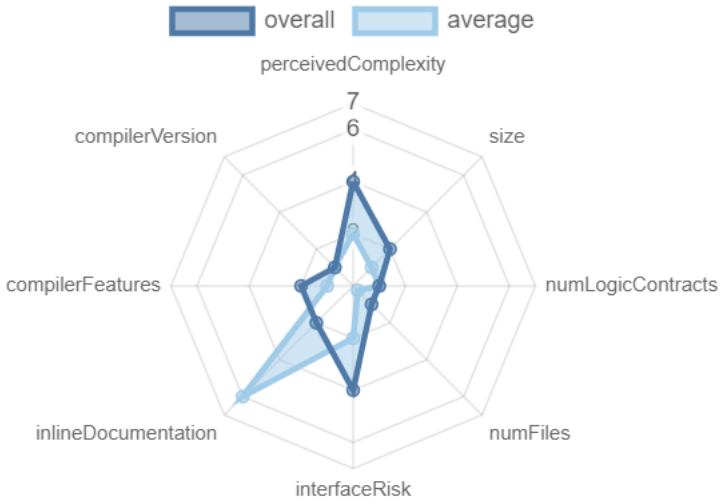
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	1	2	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	38	0

Version	External	Internal	Private	Pure	View
1.0	33	38	0	0	15

State Variables

Version	Total	Public
1.0	11	8

Capabilities

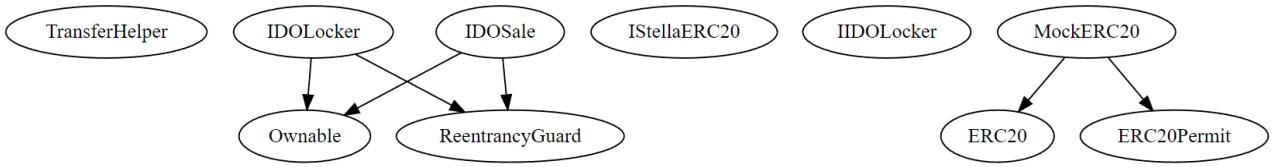
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

1.0	`>=0.6.0`
	`^0.8.0`
	`^0.8.2`
				Yes	

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

Inheritance Graph

v1.0



Call Graph

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

Is contract an upgradeable

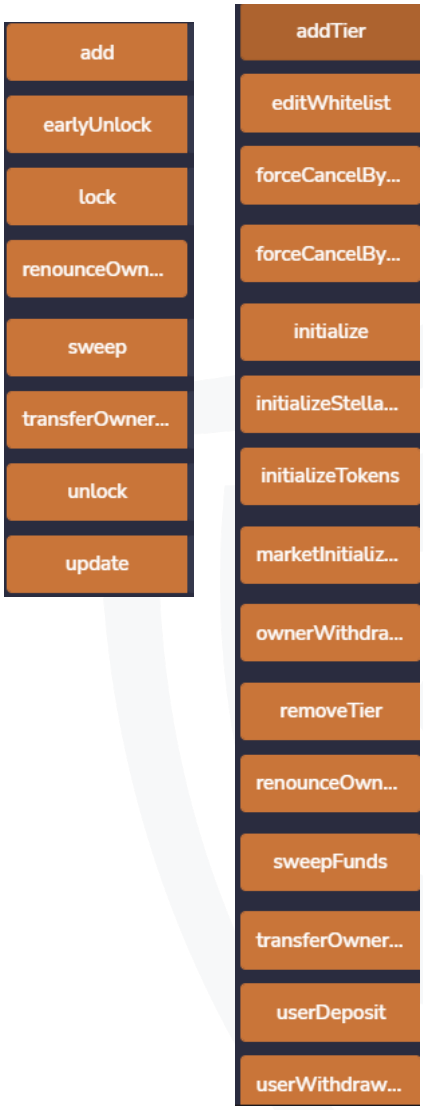
Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
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 contracts

v1.0



Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

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

```
lock                9+
nonReentrant
notContract
unlock              9+
nonReentrant
notContract
earlyUnlock         9+
nonReentrant
notContract
add                 9+
onlyOwner
update              9+
onlyOwner
sweep               9+
onlyOwner
```

```
initialize                  9+
onlyOwner
initializeStellaSettings    9+
onlyOwner
initializeTokens            9+
onlyPresaleOwner
userDeposit                 9+
userWithdrawTokens          9+
userWithdrawBaseT...        9+
ownerWithdrawTok...         9+
onlyPresaleOwner
sweepFunds                  9+
onlyOwner
editWhitelist               9+
onlyOwner
forceCancelByStella         9+
onlyOwner
forceCancelByPresal...      9+
onlyPresaleOwner
```

```
marketInitialized
onlyOwner
addTier
onlyOwner
removeTier
onlyOwner
```

Comments:

- Some of the functions has a modifier "onlyOwner" which allows one authority to do certain actions in the contract and we have also noticed that the fee of the swap is also being transferred to the owner address as well.

- The owner of the contract can cancel the sale anytime they want. Moreover the presaleOwner can also cancel the sale.

- The owner can set the lock and unlock time for the pool as same which will result in pool ending instantly after locking

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/ido/TransferHelper.sol	1		29	29	19	5	26	
	contracts/ido/IDOLocker.sol	1		217	201	157	22	86	
	contracts/ido/IDOSale.sol	1	2	376	320	257	38	175	
	contracts/utils/MockERC20.sol	1		32	32	24	1	16	
	Totals	**4**	**2**	**654**	**582**	**457**	**66**	**303**	

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

Issue	File	Type	Line	Description
#1	IDOLocker.sol	Start and End time can be equal	145	When calling the add function there is no check that will ensure that the startTime, endTime, unlockTime shouldn't be equal. If/When they will be equal then the pool will be available to unlock or will end immediately after lock
#2	IDOSale.sol	Fee can be 100 percent	138	The base and token fee can be set to more than or equal to 100 percent which will result in transferring all of the token amount set for presale by the presale owner in case owner cancels the sale and the presale owner calls the withdraw function

Low issues

Issue	File	Type	Line	Description
#1	IDOLocker.sol /IDOSale.sol	A floating pragma is set	3	The current pragma Solidity directive is „“^0.8.0"".
#2	IDOSale.sol	Missing Events	No events in the contract	Emit an event for critical parameter changes. In this case, minting, burning of tokens, etc.
#3	IDOLock.sol	Missing Events	145, 176,	Emit an event for critical parameter changes. In this case, minting, burning of tokens, etc.
#4	IDOSale.sol	Missing zero check	138, 296	Check that the address is not zero
#5	IDOLock.sol	Redundant Code	108-120 and 122	These two segments contain exactly the same code.

Informational issues

Issue	File	Type	Line	Description
#1	IDOSale.sol	Constable State Variable	102	This state variable is never modified in the contract and should be declared constant
#2	IDOSale.sol	Unused return values	296	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
#3	IDOSale.sol	Missing check for existing entries	296	While adding/removing an account from whitelist, there should be a check to see of the account already exists or not
#4	IDOLock/IDOSale.sol	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#5	IDOLock/IDOSale/TransferHelper.sol	Commented code exists	-	We recommend to remove all the dead code present in the contract

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.

28.July,2022:

- There is still an owner (Owner still has not renounced ownership)
- Developer can cancel the sale anytime.
- Owner can blacklist/whitelist any address
- There are no constructors in the contracts, initialize functions has been used that can be called multiple times.
- Read the whole report and modifiers section 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	NOT 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	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED

	character (U+202E)		
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	CWE-362: Concurrent Execution using Shared Resource with Improper	PASSED

	Dependence	Synchronization ('Race Condition')	
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 SELFDE	CWE-284: Improper Access Control	PASSED

	STRUCT Instruction		
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

SolidProof.io

@solidproof_io

SolidProofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY