SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Oscar Swap

Audit

Security Assessment
28. May, 2023

For

SolidProof_io @solidproof_io

Disclaimer

Version	Date	Description
1.0	29. April 2023 - 1. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Arbitrum

Website
https://oscarswap.com/

Medium
https://medium.com/@oscarswap

Twitter
https://twitter.com/Oscar_Swap

Discord
https://discord.gg/G8Qfn7cmjy

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

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

Description

Oscarswap is a decentralized exchange (DEX) that operates on the Arbitrum network, utilizing automated market-maker (AMM) technology. Its cutting-edge technology is designed to offer the lowest fees for swapping cryptocurrencies, coupled with highly profitable yield farming rewards, making it an ideal choice for passive income seekers.Oscarswap is a decentralized exchange (DEX) that operates on the Arbitrum network, utilizing automated market-maker (AMM) technology. Its cutting-edge technology is designed to offer the lowest fees for swapping cryptocurrencies, coupled with highly profitable yield farming rewards, making it an ideal choice for passive income seekers.

Project Engagement

During the 28 of April 2023, **OscarSwap 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 Links

v1.0

Arbitrum Network

Oscarswap Token: https://arbiscan.io/address/0xe77e0C559494585aC396A91BF35fB164E272b896

Factory: 0x20fc9D10d7391bC9C7F338fd94F7185B0Fed9A4C

Timelock: 0x2bee4903ffeebecBBd2Dac20Ef823fa566F3EBfd

Masterchef: 0x1A635bb3fC03e6e7109eBdFb61a4DA971B37A329

Multisig: 0xe8ffe751dea181025a9acf3d6bde8cda5380f53f

Router: 0x4d381C158d74c88dA251BabfE33d320239324213

Zap: 0x1627c27eB95ee0856Cc1a76484D3F5d9cBEE167c

OscarPool: 0x826fb9072Dd8C5187B0ac1D1429F7bFc9e3F93ee

OscarFlexPool: 0xC680aD5a7C42BF8a6a669D08C40317fa1a16b7bC

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.

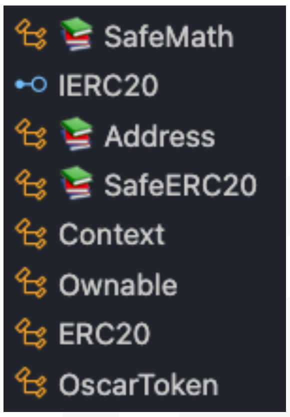
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

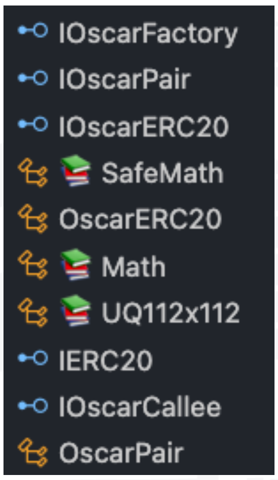
Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

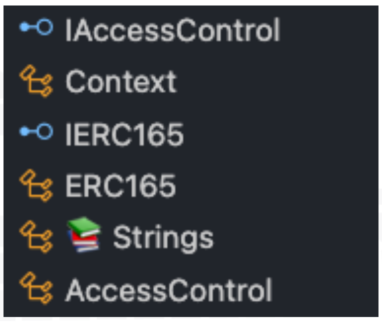
MasterChef


SafeMath
IERC20
Address
SafeERC20
Context
Ownable
ERC20
OscarToken

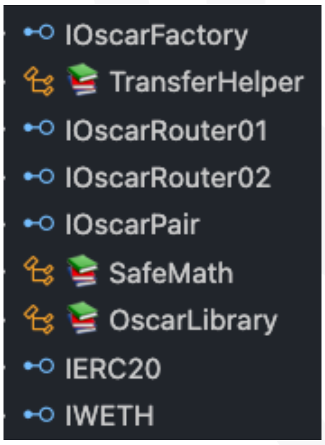
OscarFactory


IOscarFactory
IOscarPair
IOscarERC20
SafeMath
OscarERC20
Math
UQ112x112
IERC20
IOscarCallee
OscarPair

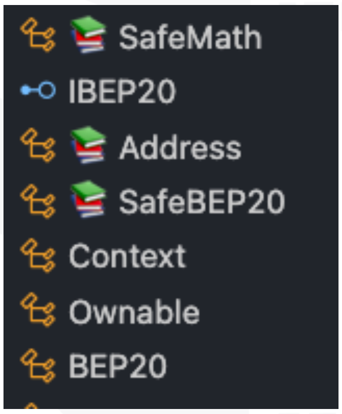
TimelockController


IAccessControl
Context
IERC165
ERC165
Strings
AccessControl

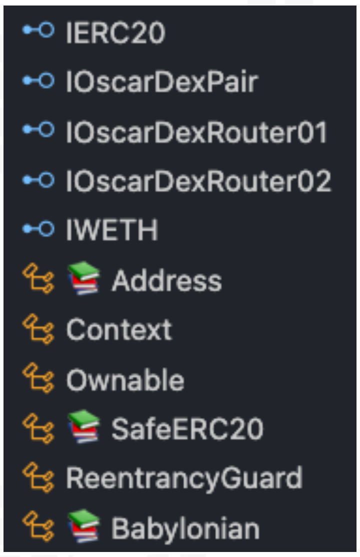
OscarRouter


IOscarFactory
TransferHelper
IOscarRouter01
IOscarRouter02
IOscarPair
SafeMath
OscarLibrary
IERC20
IWETH

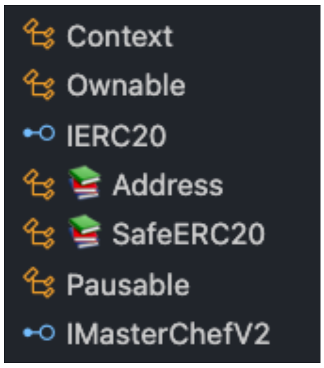
OscarToken


SafeMath
IBEP20
Address
SafeBEP20
Context
Ownable
BEP20

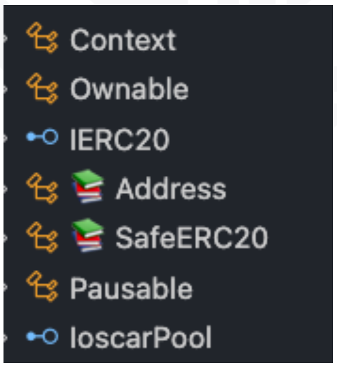
OscarDexZapV1


IERC20
IOscarDexPair
IOscarDexRouter01
IOscarDexRouter02
IWETH
Address
Context
Ownable
SafeERC20
ReentrancyGuard
Babylonian

OscarPool


Context
Ownable
IERC20
Address
SafeERC20
Pausable
IMasterChefV2

OscarFlexiblePool


Context
Ownable
IERC20
Address
SafeERC20
Pausable
IoscarPool

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.

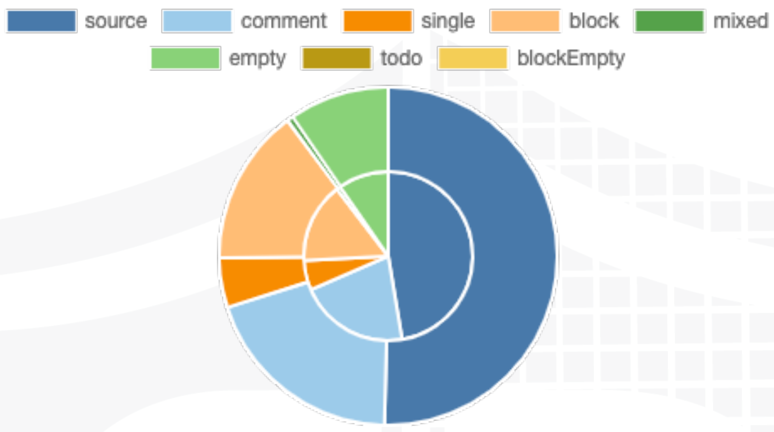
v1.0

File Name	SHA-1 Hash
contracts/TimelockController.sol	fcd24eb9c654d6fb580e0783dced1de9a438d23d
contracts/MasterChef.sol	01326904c0f9fa52ca22ef5f4fa024769c8e08aa
contracts/OscarToken.sol	a38d2b11254bce89b5ec750929c000f6da7444dd
contracts/OscarPool.sol	b82c4539861f58b65eb30c2ecbc58fbf7d974fc0
contracts/OscarFlexiblePool.sol	171459e180bbb0e422750d409334984a2c0669a1
contracts/OscarRouter.sol	38d1cdb4e71cfbe9ae81065f31be3c7b0b66333f
contracts/OscarDexZapV1.sol	154cd23ad85525ffd365c178eec84adf8be9fe78
contracts/OscarFactory.sol	f18f754e39a72d24717c8d527d53be9902e56d0d

Metrics

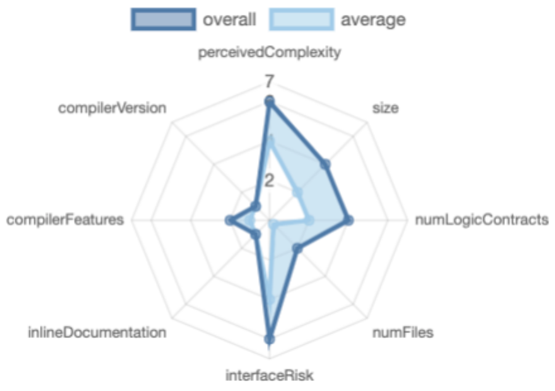
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
17	20	24	12

Exposed Functions

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

🌐Public	💰Payable
441	21

External	Internal	Private	Pure	View
329	548	19	82	189

StateVariables

Total	🌐Public
149	112

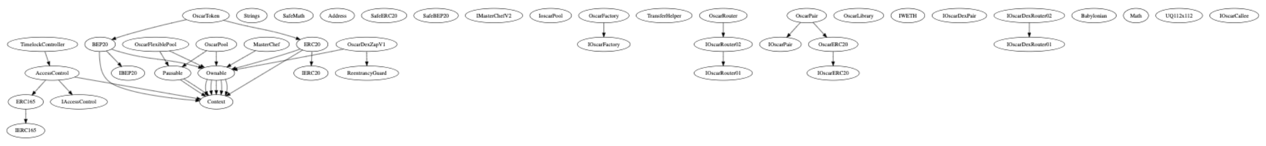
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.0` `0.8.16` `0.6.12` `=0.6.6` `>=0.5.0` `>=0.6.2` `^0.8.1` `^0.8.4` `^0.5.16`		`yes`	`yes` (12 asm blocks)	

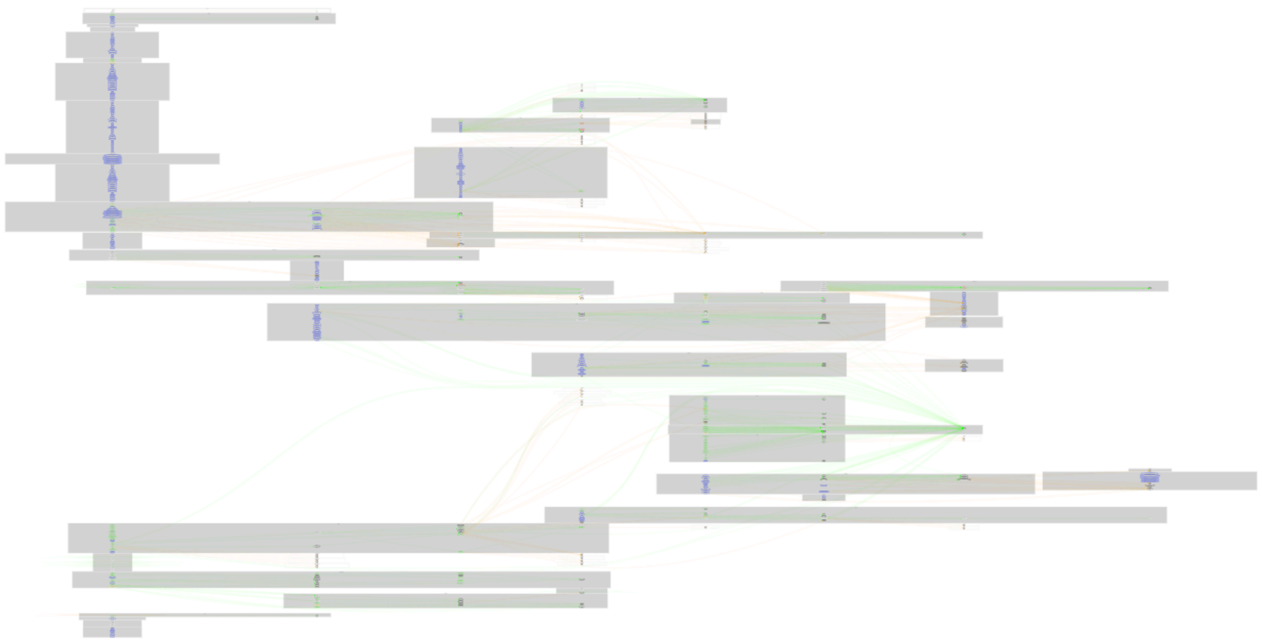
📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`		`yes`	`yes`	`yes`	`yes` → `AssemblyCall:Name:create2`

♻️ TryCatch	Σ Unchecked
	`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)

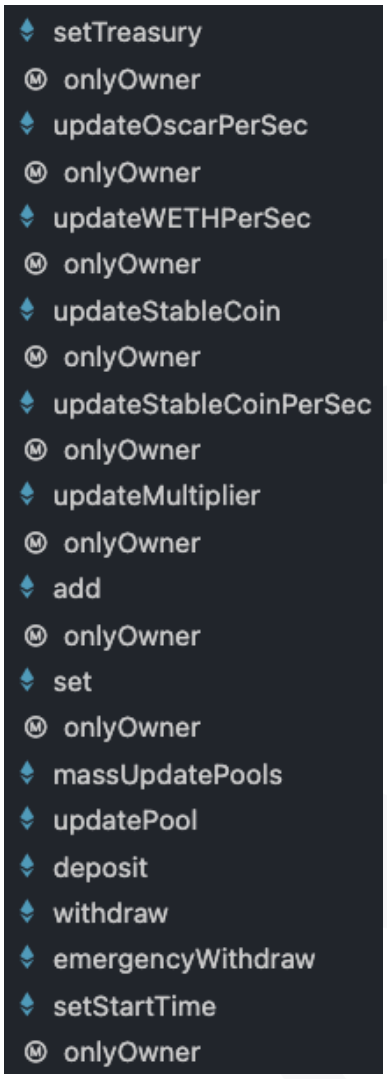
Tested	Verified
✓	✓

Legend

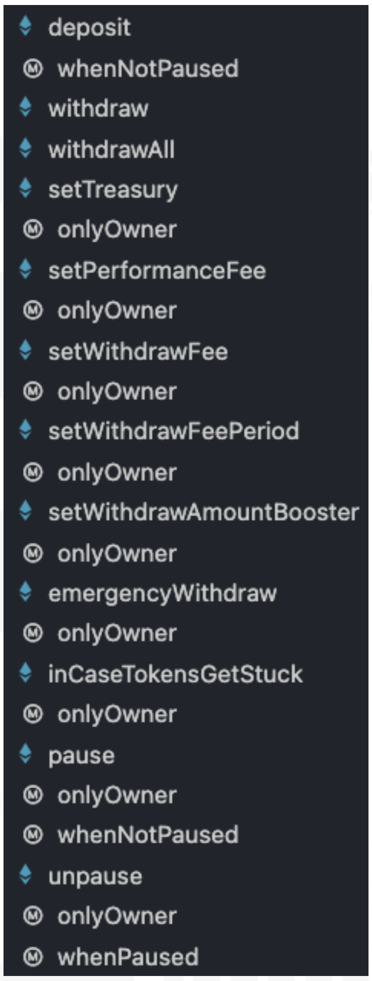
Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions
v1.0

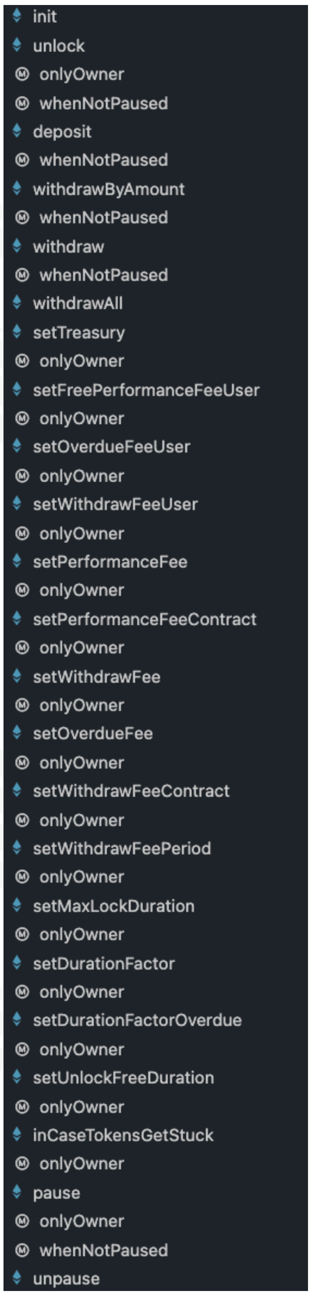
MasterChef


setTreasury
onlyOwner
updateOscarPerSec
onlyOwner
updateWETHPerSec
onlyOwner
updateStableCoin
onlyOwner
updateStableCoinPerSec
onlyOwner
updateMultiplier
onlyOwner
add
onlyOwner
set
onlyOwner
massUpdatePools
updatePool
deposit
withdraw
emergencyWithdraw
setStartTime
onlyOwner

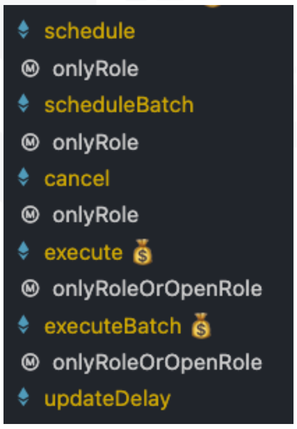
OscarFlexiblePool


deposit
whenNotPaused
withdraw
withdrawAll
setTreasury
onlyOwner
setPerformanceFee
onlyOwner
setWithdrawFee
onlyOwner
setWithdrawFeePeriod
onlyOwner
setWithdrawAmountBooster
onlyOwner
emergencyWithdraw
onlyOwner
inCaseTokensGetStuck
onlyOwner
pause
onlyOwner
whenNotPaused
unpause
onlyOwner
whenPaused

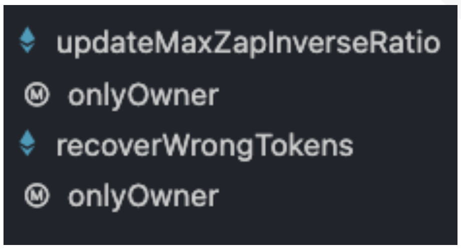
OscarPool


init
unlock
onlyOwner
whenNotPaused
deposit
whenNotPaused
withdrawByAmount
whenNotPaused
withdraw
whenNotPaused
withdrawAll
setTreasury
onlyOwner
setFreePerformanceFeeUser
onlyOwner
setOverdueFeeUser
onlyOwner
setWithdrawFeeUser
onlyOwner
setPerformanceFee
onlyOwner
setPerformanceFeeContract
onlyOwner
setWithdrawFee
onlyOwner
setOverdueFee
onlyOwner
setWithdrawFeeContract
onlyOwner
setWithdrawFeePeriod
onlyOwner
setMaxLockDuration
onlyOwner
setDurationFactor
onlyOwner
setDurationFactorOverdue
onlyOwner
setUnlockFreeDuration
onlyOwner
inCaseTokensGetStuck
onlyOwner
pause
onlyOwner
whenNotPaused
unpause

TimelockController


schedule
onlyRole
scheduleBatch
onlyRole
cancel
onlyRole
execute 💰
onlyRoleOrOpenRole
executeBatch 💰
onlyRoleOrOpenRole
updateDelay

OscarDexZapV1


updateMaxZapInverseRatio
onlyOwner
recoverWrongTokens
onlyOwner

Note:

❖ General fork from PancakeSwap

- Contracts inside are the same as the pancake-smart-contracts directory
    - https://github.com/pancakeswap/pancake-smart-contracts/tree/master/projects
    - Differences between OscarSwap and PancakeSwap contracts are the following:
        - MasterChefv2 contract does not have the staking functionality as it does in the Pancake swap. And an added stable coin reward mechanism along with the native token
        - Boost Weight functionality of CakePool has been removed from OscarPool
        - Factory contract has no changes that are critical to the logic
        - OscarRouter has no modified functionalities

Ownership Privileges

❖ *MasterChef.sol* -

- Set Treasury and StableCoin address
- Update OscarPerSec, WETHPerSec, StableCoinPerSec, and Multiplier to any arbitrary value.
- Add new Lp to the pool
- Set allocation point for a given ARX
- Set Start Time

❖ *OscarPool.sol* -

- Unlock user Oscar funds only when the contract is not paused
- Pause/Unpause Deposits and Withdraws
- Set treasury address, fee address, overdue fee address, and withdraw fee address
- Set performance fee but cannot be set more than 20%
- Set withdraw fee but cannot be set more than 5%
- The overdue fees can be set up to 100%. This fee will only be levied based on users overdue duration. Beware of it
- Set withdraw fee contract and period
- Set max lock duration but not more than 1000 days, but keep in mind that it could be set to zero.
- Set duration factor, duration factor overdue, and unlock free duration
- Withdraw unexpected tokens from the contact, but not the staked one

- ❖ *OscarFlexiblePool.sol*
  - Pause/Unpause Deposits
  - Set treasury address
  - Set performance, and withdraw fee
  - Set withdraw fee period and amount booster
  - Owner can withdraw their shares while the staking is true, and once it is done then no more staking could be done in the contract.
  - Withdraw unexpected tokens from the contract but not the deposit tokens

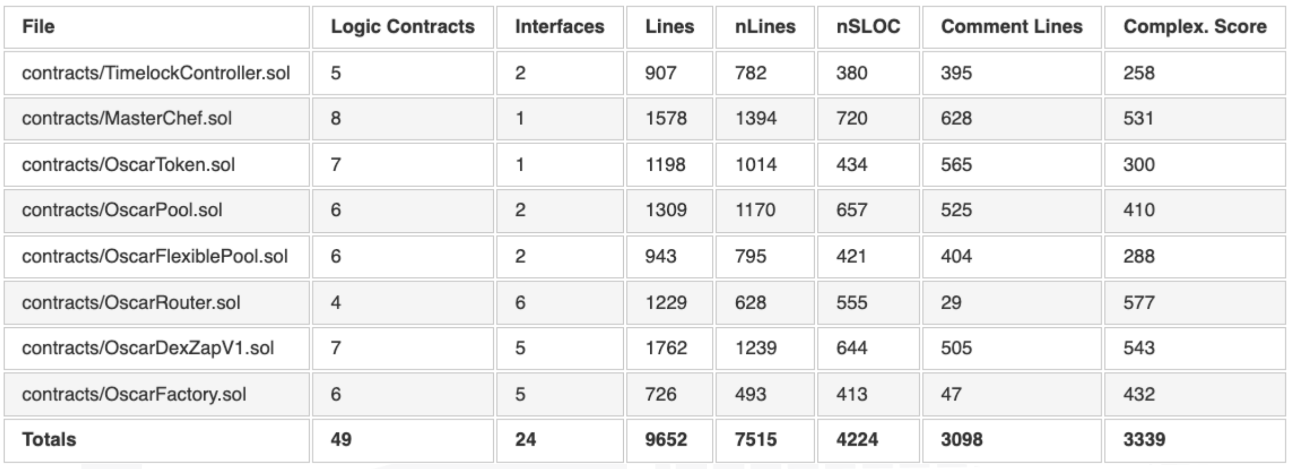
- ❖ *OscarToken.sol*
  - Owner can Mint tokens but not more than max supply which is 50 Million

- ❖ *TimelockController.sol*
  - The Proposer Role contract/wallet can schedule, cancel, and execute an operation

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

Source Units in Scope

v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/TimelockController.sol	5	2	907	782	380	395	258
contracts/MasterChef.sol	8	1	1578	1394	720	628	531
contracts/OscarToken.sol	7	1	1198	1014	434	565	300
contracts/OscarPool.sol	6	2	1309	1170	657	525	410
contracts/OscarFlexiblePool.sol	6	2	943	795	421	404	288
contracts/OscarRouter.sol	4	6	1229	628	555	29	577
contracts/OscarDexZapV1.sol	7	5	1762	1239	644	505	543
contracts/OscarFactory.sol	6	5	726	493	413	47	432
Totals	**49**	**24**	**9652**	**7515**	**4224**	**3098**	**3339**

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	All	Multiple pragma is set	—	Some of the contracts contain different pragma versions which is not recommended for deployment. We recommend to have the same pragma in all contracts and also to update the old pragma versions to the new ones.
#2	MasterChefv2.sol	Missing Zero Address Validation (missing-zero-check)	1307	Check that the address is not zero
#3	MasterChef.sol	Missing Events Arithmetic	13017-1328, 1338, 1361	Emit an event for critical parameter changes
#4	OscarRouter.sol	Old Compiler Version	2	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities

#5	OscarFactory.sol	Old Compiler Version	2	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities
#6	OscarToken.sol	Old Compiler Version	3	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities

Informational issues

Issue	File	Type	Line	Description
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities

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.

28. May 2023:

- This project consists of the following forks
  - Pancake Swap
  - Uniswap
- Unit tests with 100% code coverage was not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend OscarSwap team to conduct unit and fuzz tests thoroughly to rule out possibilities of an unwanted logical and calculation errors.
- Read whole report and modifiers section for more information
- The low issues that exist in the PancakeSwap codebase still exist in the forked code.
- We recommend using a multisig wallet for the owner address to prevent any risk of the loss of private key
- Do your own research here

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	**NOT 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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY