SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

IceCreamSwap - Launchpad

Audit

Security Assessment 13. June, 2023

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	13. June 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Bitgert
Core
XDC
Binance smart chain (only bridge deployed for now)
Dogechain
Fuse

Website

https://icecreamswap.com/?chainId=1116

Telegram

https://t.me/Icecreamswap_com

Twitter

https://twitter.com/icecream_swap

Description

Trade, Earn, Bridge and Launch on CORE, XDC, Binance smart chain (BSC), Bitgert (Brise), Shardeum, Dogechain, Doken and Fuse with our decentralized smart contracts.

Project Engagement

During the 26th of May 2023, **IceCreamSwap 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

- https://github.com/IceCreamSwapCom/IceCreamSwap-smart-contracts/tree/master/projects/launchpad/contracts
- Commit: da446c3fee322e3d57d540d572f82a2a04daeb34

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.

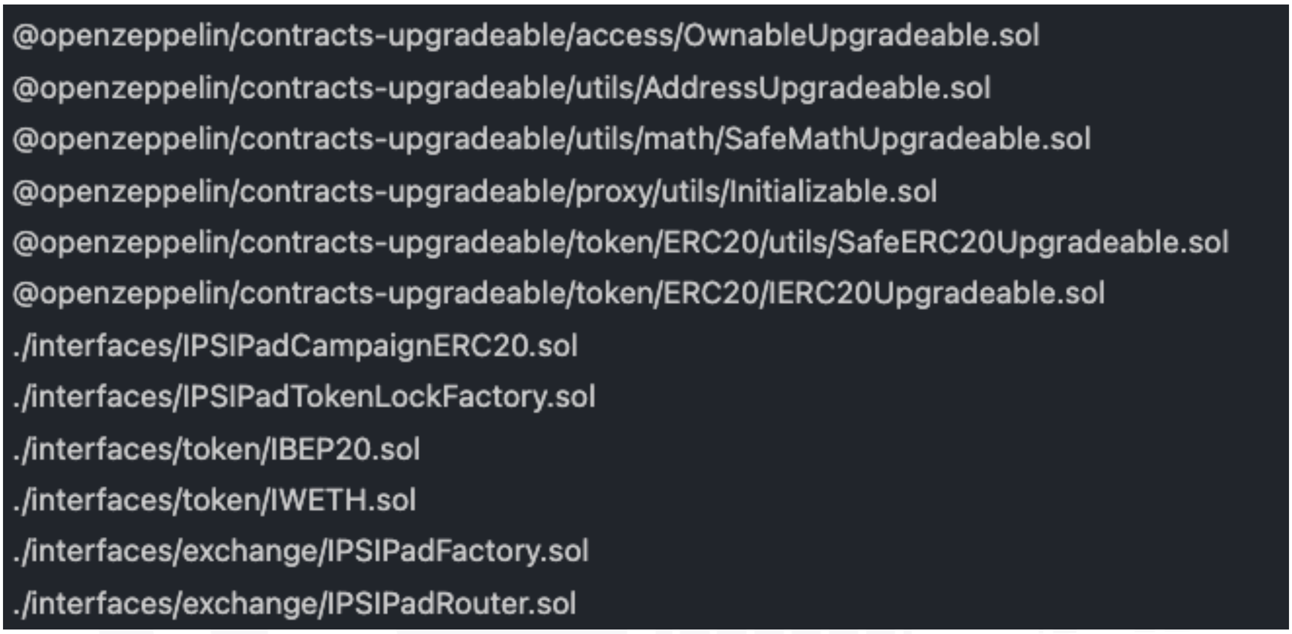
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



```
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol
./interfaces/IPSIPadCampaignERC20.sol
./interfaces/IPSIPadTokenLockFactory.sol
./interfaces/token/IBEP20.sol
./interfaces/token/IWETH.sol
./interfaces/exchange/IPSIPadFactory.sol
./interfaces/exchange/IPSIPadRouter.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

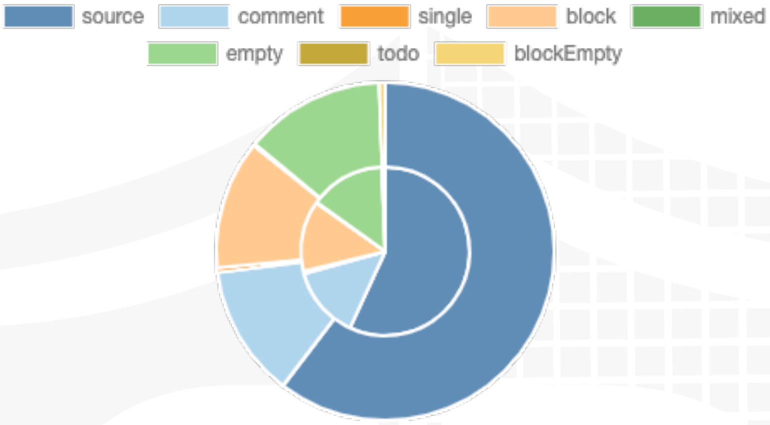
File Name	SHA-1 Hash
launchpad/contracts/interfaces/IFeeAggregator.sol	490935150b2795acc9efe5f1a0cc7070f600d22e
launchpad/contracts/interfaces/IPSIPadTokenDeployer.sol	f5f0220a2bb108626b6ec722c2e148a0167d40d6
launchpad/contracts/interfaces/IPSIPadCampaignFactory.sol	41081bdb8d9203777a968f404968786567442c98
launchpad/contracts/interfaces/IPSIPadCampaignERC20.sol	949082301e8ef5056daea0e14fa15340a6748d1e
launchpad/contracts/interfaces/IPSIPadCampaign.sol	f2d0c8f86d0b872b128c6dfc05e569dc348e2125
launchpad/contracts/interfaces/IPSIPadTokenLockFactory.sol	68e7c2f73fb6ac3a2747b743d59ed5d327d81a47
launchpad/contracts/PSIPadCampaignTrustedERC20Standalone.sol	0614b333249cb46d239f8b5289f4bd807a3c4ec0
launchpad/contracts/PSIPadTokenLockFactory.sol	2fcfc060559ca40fecc71e1d41486a52c5fb01b2
launchpad/contracts/interfaces/exchange/IPSIPadRouter.sol	74c8e40df02fe84ff8f53d385c33acc5a4ddcb71
launchpad/contracts/interfaces/exchange/IPSIPadFactory.sol	bb8478f1204ad0f7c8121e252a0626e5e9e868c5
launchpad/contracts/interfaces/token/IERC2612.sol	da842b50d0c988830df7b17d6e3d0be9b7b6c984

launchpad/contracts/interfaces/token/IWETH.sol	4777a15c81aa5e94d9e6c7c9d83e2eb1a99f2efc
launchpad/contracts/interfaces/token/IBEP20.sol	dd45f777454eee735d4586542edcfc7d6e8d7946
launchpad/contracts/interfaces/token/IERC677.sol	7147e3c8d6ba60cf865cfdc72068408f268f40cb
launchpad/contracts/interfaces/token/crosschain/IAnyswapV4ERC20.sol	ef4d6e60e60352409945e934f252bdd644cae394

Metrics

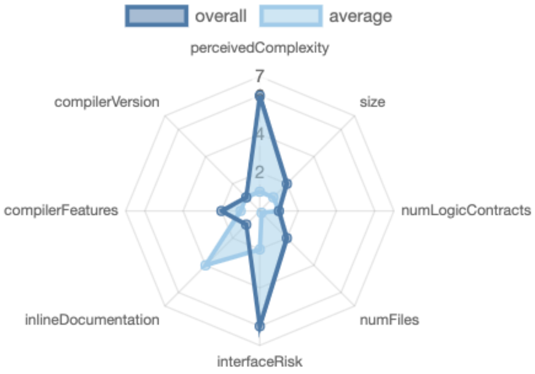
Source Lines

v1.0



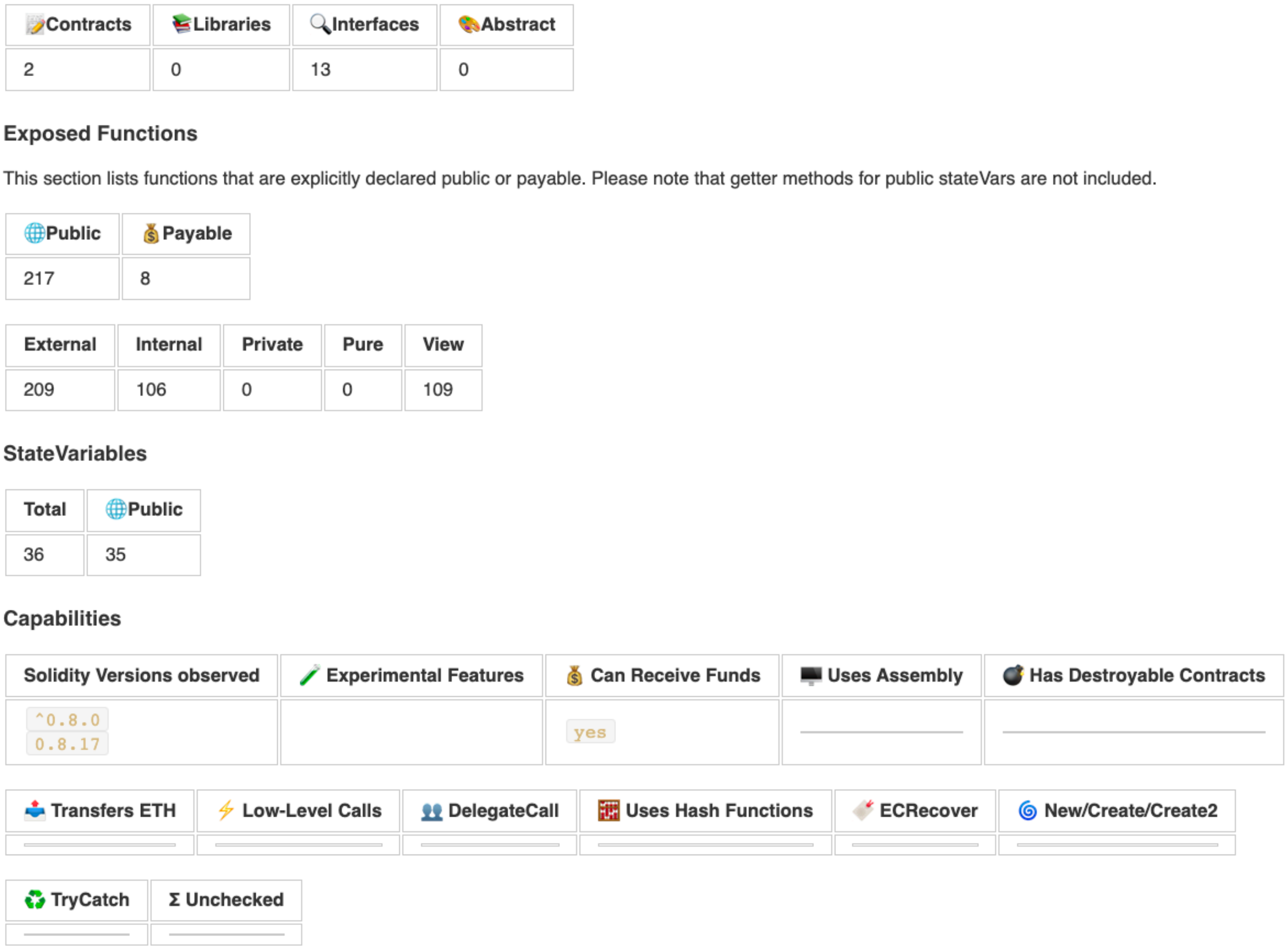
Risk Level

v1.0



Capabilities

Components



📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
2	0	13	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.

🌐Public	💰Payable
217	8

External	Internal	Private	Pure	View
209	106	0	0	109

StateVariables

Total	🌐Public
36	35

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

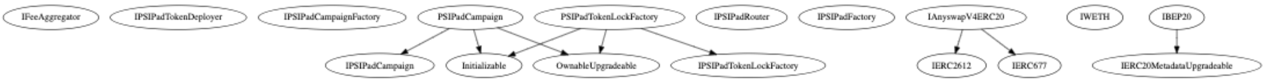
`^0.8.0`
`0.8.17`
		`yes`		

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2

♻️ TryCatch	Σ Unchecked

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. Is contract an upgradeable
2. Deployer cannot lock user funds
3. Deployer cannot pause the contract
4. Deployer cannot set fees
5. Deployer cannot blacklist/antisnipe addresses
6. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the contracts which can change any limit and give owner new privileges
    - Be aware of this and do your own research for the contract which is the contract pointing to

Deployer cannot lock user funds

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

Comments:

v1.0

- The owner can lock user funds by
    - blacklisting addresses
    - Changing the end date

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can pause	✓	✓	✗

Comments:

v1.0

- Owner can pause the contract

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer can set fees over 25%	✓	✓	✗
Deployer can set fees to nearly 100% or to 100%	✓	✓	✗

Comments:

v1.0

- Fees can be set without any limitations

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- Owner can whitelist/blacklist addresses

Overall checkup (Smart Contract Security)

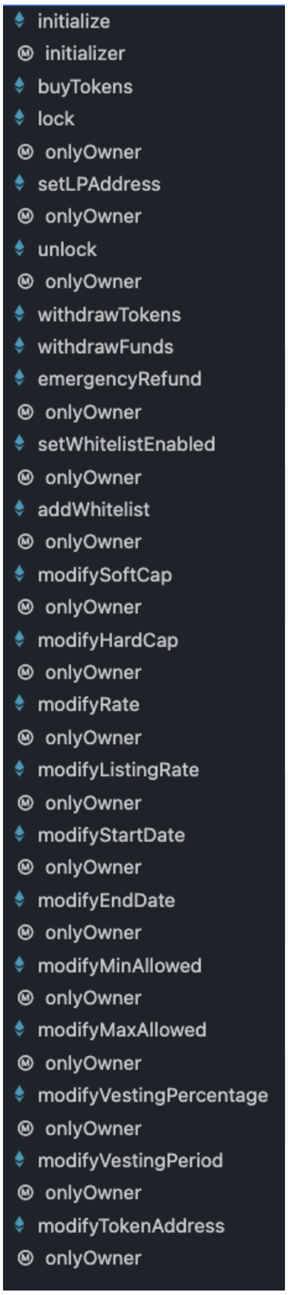
Tested	Verified
✓	✓

Legend

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

Modifiers and public functions

v1.0



```
♦ initialize
Ⓜ initializer
♦ buyTokens
♦ lock
Ⓜ onlyOwner
♦ setLPAddress
Ⓜ onlyOwner
♦ unlock
Ⓜ onlyOwner
♦ withdrawTokens
♦ withdrawFunds
♦ emergencyRefund
Ⓜ onlyOwner
♦ setWhitelistEnabled
Ⓜ onlyOwner
♦ addWhitelist
Ⓜ onlyOwner
♦ modifySoftCap
Ⓜ onlyOwner
♦ modifyHardCap
Ⓜ onlyOwner
♦ modifyRate
Ⓜ onlyOwner
♦ modifyListingRate
Ⓜ onlyOwner
♦ modifyStartDate
Ⓜ onlyOwner
♦ modifyEndDate
Ⓜ onlyOwner
♦ modifyMinAllowed
Ⓜ onlyOwner
♦ modifyMaxAllowed
Ⓜ onlyOwner
♦ modifyVestingPercentage
Ⓜ onlyOwner
♦ modifyVestingPeriod
Ⓜ onlyOwner
♦ modifyTokenAddress
Ⓜ onlyOwner
```

Ownership Privileges

❖ *PSIPadCampaignTrustedERC20Standalone.sol* -

- The owner can add liquidity and burn the remaining tokens when the liquidity is not locked.
- Unlock the LP tokens
- Enable or Disable the whitelist
- Add/Remove addresses from the whitelist.
- Modify the following parameters at any time without any limitations, even after the campaign is live, so this gives the owner

every possibility to control every aspect of the campaign mentioned below:

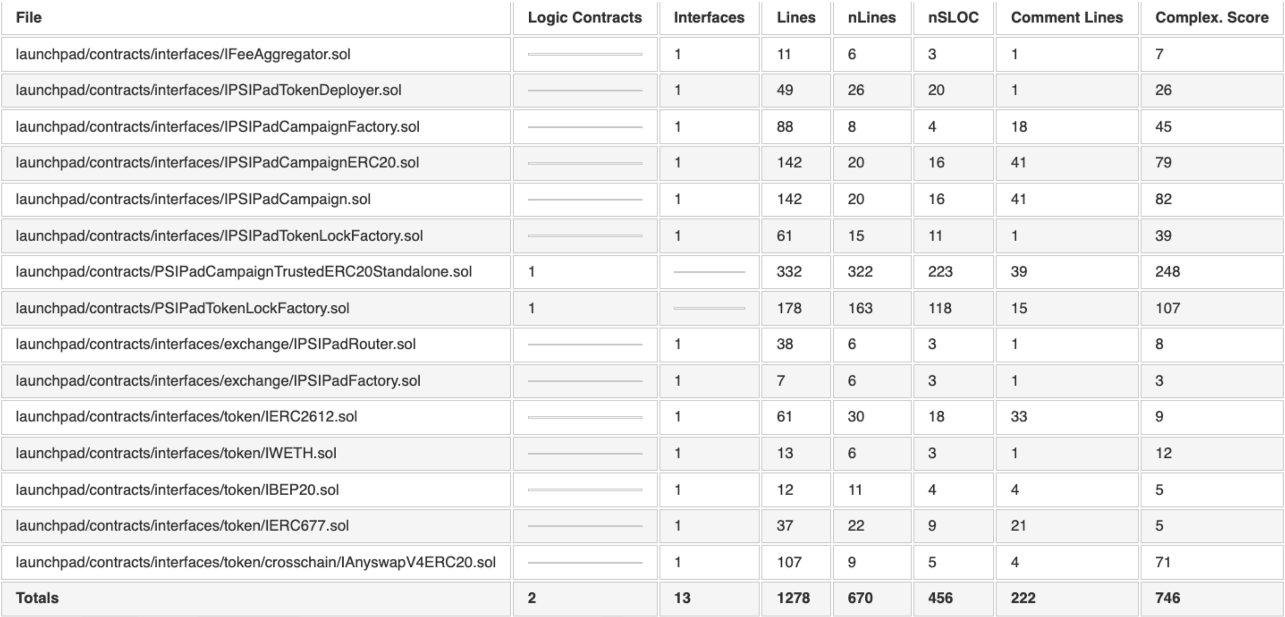
- Hard Cap, Soft Cap
- Price and Listing Rate
- Start and End date
- Modify the minimum and maximum amount of tokens that are allowed to be bought. Moreover setting it to zero will result in the pause of the contract’s functionality
- Modify the Vesting Percentage and Period to any value
- Modify token address

- *PSIPadFactory.sol* -
  - Change fee aggregator, wrapped coin address
  - Set wrapped coin fee to any arbitrary value including 100% or more which is not recommended
  - Owner can unlock tokens and then the unlocked tokens will be transferred to the owners account

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
launchpad/contracts/interfaces/IFeeAggregator.sol		1	11	6	3	1	7
launchpad/contracts/interfaces/IPSIPadTokenDeployer.sol		1	49	26	20	1	26
launchpad/contracts/interfaces/IPSIPadCampaignFactory.sol		1	88	8	4	18	45
launchpad/contracts/interfaces/IPSIPadCampaignERC20.sol		1	142	20	16	41	79
launchpad/contracts/interfaces/IPSIPadCampaign.sol		1	142	20	16	41	82
launchpad/contracts/interfaces/IPSIPadTokenLockFactory.sol		1	61	15	11	1	39
launchpad/contracts/PSIPadCampaignTrustedERC20Standalone.sol	1		332	322	223	39	248
launchpad/contracts/PSIPadTokenLockFactory.sol	1		178	163	118	15	107
launchpad/contracts/interfaces/exchange/IPSIPadRouter.sol		1	38	6	3	1	8
launchpad/contracts/interfaces/exchange/IPSIPadFactory.sol		1	7	6	3	1	3
launchpad/contracts/interfaces/token/IERC2612.sol		1	61	30	18	33	9
launchpad/contracts/interfaces/token/IWETH.sol		1	13	6	3	1	12
launchpad/contracts/interfaces/token/IBEP20.sol		1	12	11	4	4	5
launchpad/contracts/interfaces/token/IERC677.sol		1	37	22	9	21	5
launchpad/contracts/interfaces/token/crosschain/IAnyswapV4ERC20.sol		1	107	9	5	4	71
Totals	**2**	**13**	**1278**	**670**	**456**	**222**	**746**

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

Medium issues found

Issue	File	Type	Line	Description	Status
#1	PSIPadTokenLockFactory.sol	Fees can be any arbitrary value	73	The owner can set the fees to any arbitrary value which can result in users not able to lock tokens because of the very high fees. We recommend putting a hard cap on the fees.	Open
#2	PSIPadTokenLockFactory.sol	Wrapped_coin	121-125	The wrapped_coin can be set to zero or dead address. The owner can lock the "lock" function if the fee's are above 0.	Open
#3	PSIPadCampaignTrustedERC20Standalone.sol	raisedToken, factory_address, router_address, lock_address	82, 95, 96, 98	The raisedToken cannot be updated that causes if the token is accidentally set to zero/dead the following functions will not work: - buyTokens - addLiquidity - Unlock - withdrawFunds	Open
#4	PSIPadCampaignTrustedERC20Standalone.sol	Enabling	110	The owner is able to enable/disable buying tokens functions. In this case the owner can allow addresses to buy tokens and revert the whiteliste when the trading is disabled. Addresses that bought tokens are not able to tr	Open

#5	PSIPadCampaignTrustedERC20Standalone.sol	Centralized	See description	The owner is able to change any variables while the campaign is live. That causes that the owner is able to manipulate the campaigns while it is ongoing.	Open

Low issues

Issue	File	Type	Line	Description	Status
#1	PSIPadTokenLockFactory.sol	A floating pragma is set	—	The current pragma Solidity directive is „“^0.8.0”.	Open
#2	PSIPadTokenLockFactory.sol	Missing Zero Address Validation (missing-zero-check)	65, 69, 56-58	Check that the address is not zero	Open
#3	PSIPadCampaignTrustedERC20Standalone.sol	Missing Zero Address Validation (missing-zero-check)	67-74, 330	Check that the address is not zero	Open
#4	PSIPadCampaignTrustedERC20Standalone.sol	Missing Events Arithmetic	All	Emit an event for critical parameter changes	Open

#5	PSIPadTokenLockFactory.sol	Missing Events Arithmetic	All	Emit an event for critical parameter changes	Open
#6	PSIPadCampaignTrustedERC20Standalone.sol	Min_allowed can be zero	314	If the min_allowed variable is set to 0, the condition "(hardCap - collected) < min_allowed" L120 will never be 0 because the hardcap and collected are "uint256" type variables. It can never below 0.	Open
#7	PSIPadCampaignTrustedERC20Standalone.sol	Vesting percentage can be set over 100%	322	It is recommended to check the percentage is not over 100% because of the lockAmount in L211	Open

Informational issues

Issue	File	Type	Line	Description	Status
#1	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Open
#2	PSIPadTokenLockFactory	Unnecessary Safemath	Look into contract for safemath functions	The safemath library is unnecessary because it is handled by pragma version above 0.8.x by default. Remove safemath functionalities and replace them with raw mathematical operations.	Open

#3	PSIPadCampaignTrustedERC20Standalone.sol	Amount is zero check	208	It is recommended to check if the amount is zero.	Open
#4	PSIPadCampaignTrustedERC20Standalone.sol	tokenLiquidity is zero check	153	It is recommended to check if the amount is zero because it can be set to 0.	Open
#5	PSIPadCampaignTrustedERC20Standalone.sol	Misspelling	See descritpion	Adjusting the following misspellings is recommended: - liqudity L127	Open

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. June 2023:

- There is still an owner (The owner still has not renounced ownership)
- The contracts are completely centralized, and the owner can change every parameter
- 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	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

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY