



**SOLID**Proof  
*Bring trust into your projects*

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

MADE IN GERMANY

# **IceCreamSwap** Staking **Audit**

**Security Assessment**  
**14. June, 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	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	17
Source Units in Scope	18
Critical issues	19
High issues	19
Medium issues	19
Low issues	20
Informational issues	20
Audit Comments	21
SWC Attacks	22

# 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	01. June 2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## 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](https://t.me/Icecreamswap_com)

## Twitter

[https://twitter.com/icecream\\_swap](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

- Dex
  - <https://github.com/IceCreamSwapCom/IceCreamSwap-smart-contracts/tree/master/projects/staking/contracts>
  - Commit: a298754007a1fb98b5137e91954decc1015f6424

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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/access/Ownable.sol	2
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	2
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1



## Tested Contract Files

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

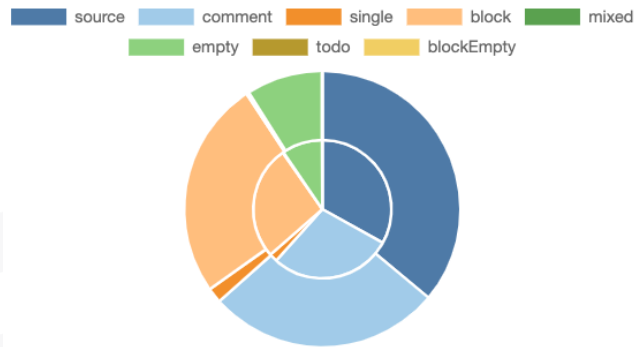
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

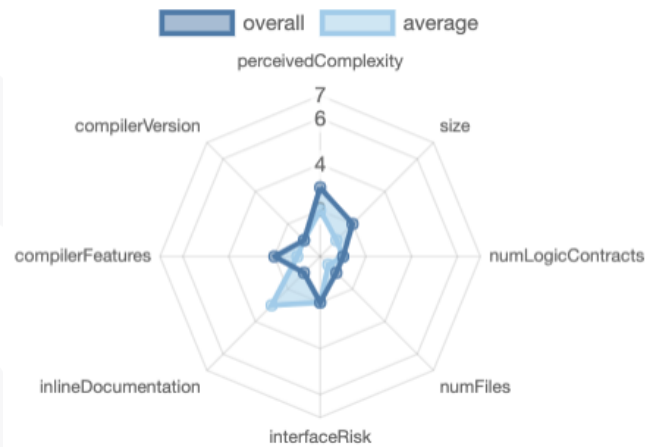
File Name	SHA-1 Hash
staking/contracts/ SmartChefInitializable.sol	4355eba5a03f2d0bdfdcc48f4653ef3 266f4a891
staking/contracts/ SmartChefFactory.sol	576c0489ff21bf271d52dbfe7905c8c 385b0e71c

# Metrics

## Source Lines v1.0



## Risk Level v1.0




# Capabilities

## Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	0	0	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
13	0







External	Internal	Private	Pure	View
12	22	0	0	3


### StateVariables

Total	 Public
15	15

### Capabilities

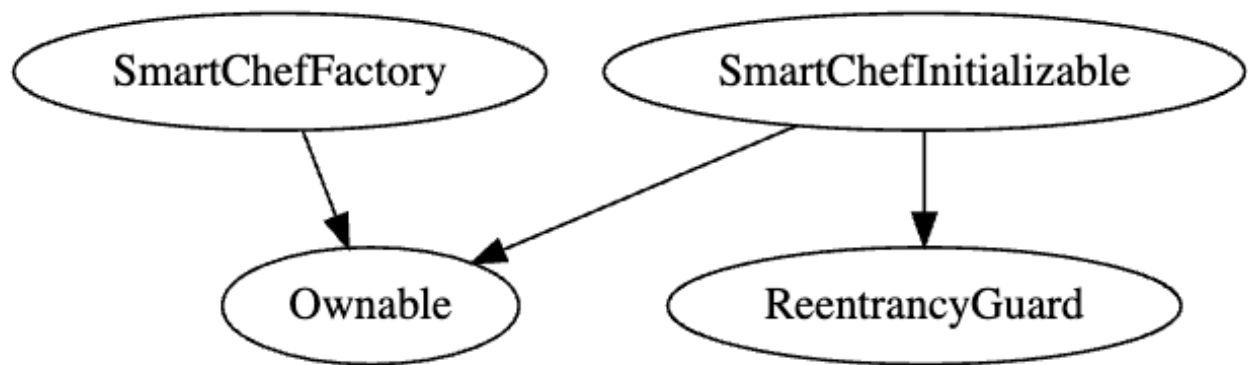
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.17			yes (1 asm blocks)	

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

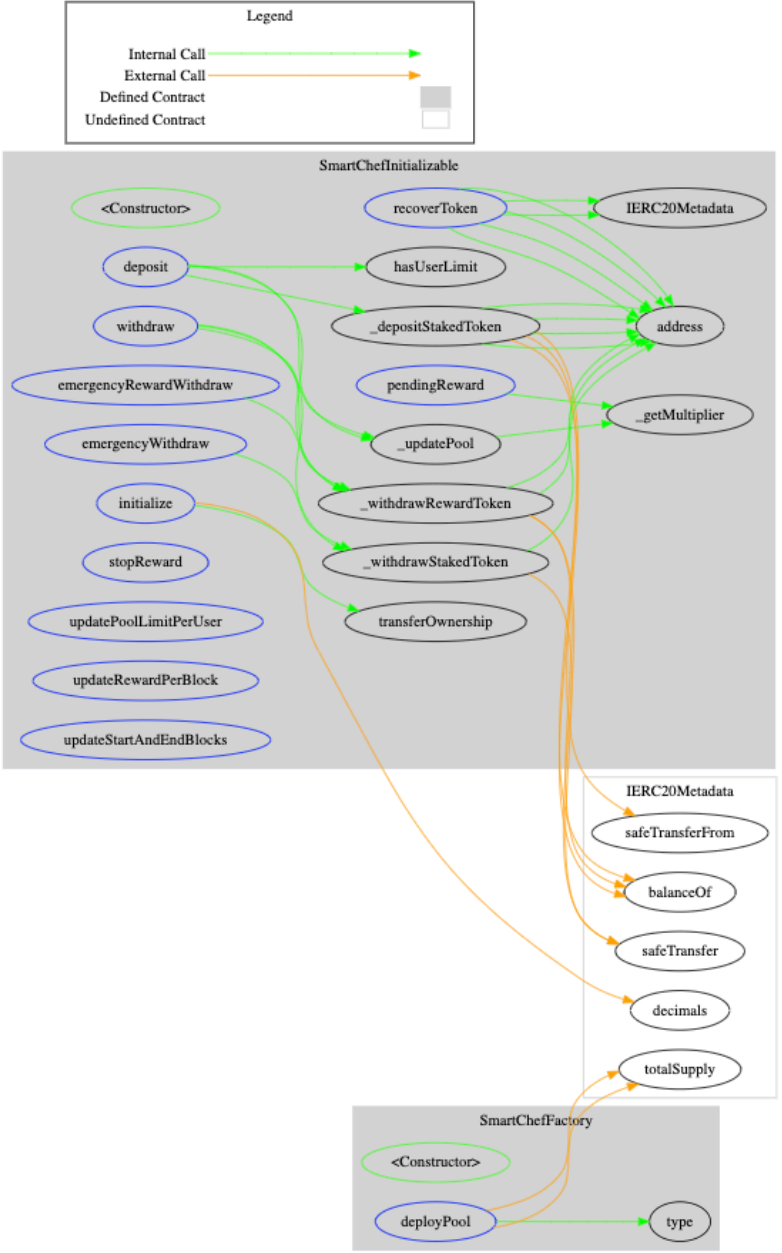
 TryCatch	 Σ Unchecked

## 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. Overall checkup (Smart Contract Security)



## Is contract an upgradeable

Name	
Is contract an upgradeable?	No



## 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

The modifiers and public functions are the same as the Pancake Swap Smart Chef Contracts found at —

<https://github.com/pancakeswap/pancake-smart-contracts/blob/master/projects/farms-pools/contracts/SmartChefInitializable.sol>

The only differences are:

- The withdrawal/deposit of staked and reward tokens are being handled by separate functions

**Note-** The owner of the “SmartInitializable” contract can withdraw all reward tokens from the contract balance. Beware of it.

**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
staking/contracts/SmartChefInitializable.sol	1	————	367	358	197	94	127
staking/contracts/SmartChefFactory.sol	1	————	65	56	32	15	32
<b>Totals</b>	<b>2</b>	————	<b>432</b>	<b>414</b>	<b>229</b>	<b>109</b>	<b>159</b>

### 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	SmartChef Initialize.sol	Missing Timelock	164	We strongly recommend to put a Timelock in place so that the funds are not available to be withdrawn immediately after deposit.	Open
#2	SmartChef Initialize.sol	Missing zero address check	92, 93	It is recommended to check the addresses “_stakedToken” and “_rewardToken” they are not zero address. After the initializing you are not able to change these addresses.	Open
#3	SmartChef Initialize.sol	Missing checks	110	<p>While initializing the startBlock can be set below block.number directly. In this case you are not able to update the Start anymore because of the checks in L274-276.</p> <p>Make sure to check the Startblock etc in the initialize function also. Also check that the Startblock is below the bonusEndBlock.</p>	Open

## Low issues

Issue	File	Type	Line	Description	Status
#1	SmartChef Initializable.sol	User info not removed	164, 190	<p>The user information from the contract is not removed after the user has withdrawn all deposited tokens then the user info must be removed otherwise in future when the list of users is very long unnecessary data storage may lead to gas issues.</p> <p>We recommend removing the users once they have no active stake in the contract</p>	Open
#2	SmartChef Initializable.sol	Check for zero value	42	<p>It is recommended to check for 0 value because the contract calculates the cakereward with this "rewardPerBlock" variable. Everything will be 0.</p>	Open
#3	SmartChef Initializable.sol	Setting "userLimit"	244-254	<p>The "userLimit" can only be set to false directly but never to true.</p> <p>It will only be set to true while initializing.</p> <p>Additionally, the userLimit will be set while depositing all the time.</p>	Open
#4	SmartChef Initializable.sol	Revert	See description	<p>When the user.rewardDebt is higher than the PRECISION_FACTOR every calculation will be reverted because of the underflow issue. Also if the calculation of "PRECISION_FACTOR - user.rewardDebt" is 0 it will be reverted because of dividing by 0 is not allowed.</p>	Open

## 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/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 variable, functions etc. do.

### 14. June 2023:

- Read the whole report and modifiers section for more information
- If some other low issues that exist in the Pancake Swap codebase are also present in the IceCream Swap Staking Contracts.
- The constructor in the “SmartChefFactory” contract appears to be unimplemented. We recommend finishing writing the code in the constructor or removing it if necessary.

## SWC Attacks

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

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

*Solid  
Proofed*

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

  
MADE IN GERMANY



<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>