SOLIDProof
Bring trust into your projects
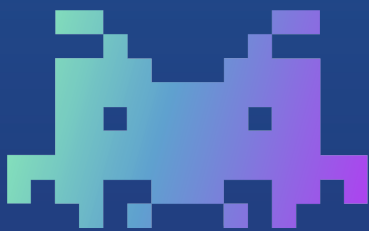
Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

BitCrush

Audit

Security Assessment
09. March, 2022

For

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	04. March 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	07. March 2022	Reaudit
1.2	09. March 2022	Reaudit
1.3	01. April 2022	Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://www.bitcrush.com/

Telegram
https://t.me/Bcarcadechat

Twitter
https://twitter.com/bitcrusharcade

Description

Bitcrush Arcade is a gaming platform aimed to further integrate the world of Defi and Gaming. It is our belief as technologies progress, the line between iGaming and decentralized finance will continue to blur.

Project Engagement

During the 3rd of March 2022, **Bitcrush 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/Bitcrush-Arcade/crush_contracts/tree/galacticFarms/contracts
  - Commit: 65a5bf7247e78e52dd0a9afab38a76d492fa15f8

v1.1

- Github
  - https://github.com/Bitcrush-Arcade/crush_contracts/tree/galacticFarms/contracts
  - Commit: b40d89005b82b0ba4e9726fe7b6b920ce07219ae

v1.2

- Github
  - https://github.com/Bitcrush-Arcade/crush_contracts/tree/galacticFarms/contracts
  - Commit: 2f15f7b132d275e28084f7a3fc6848f6176c427f

v1.3

- Github
  - FeeDistributorV3

- https://github.com/Bitcrush-Arcade/crush_contracts/blob/pythonMigration/contracts/FeeDistributorV3.sol
    - Commit: f51559841e1fe2335aaca5ff79b181083efd2f49
- InvaderPool
    - https://github.com/Bitcrush-Arcade/crush_contracts/blob/pythonMigration/contracts/InvaderPool.sol
    - Commit: b75adb6a1b6ec24138992747c9fddec0e64ae8ff
    -

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	3
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	3
@openzeppelin/contracts/utils/math/SafeMath.sol	1

@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.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
contracts/GalacticChef.sol	a649b64a30c2941d323ef92e1e8564c276774fc9
contracts/IStaking.sol	3b7ffdc51969a81503261458b1ec30418f707471
contracts/IFeeDistributor.sol	c7c8197683564b2df2302f66ab39ffa211c30001
contracts/FeeDistributor.sol	69f9c29f339ecb30a270a9afc325ad6843c81abc
contracts/niceStaking.sol	5bffde5e01ca9510e8eb3256ad914ab48eff872e

v1.2

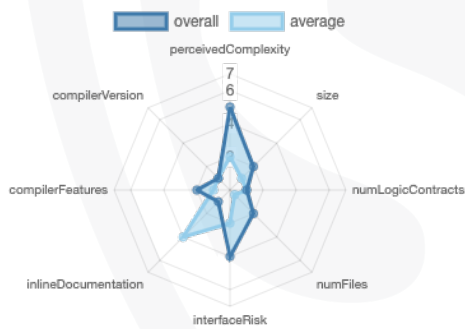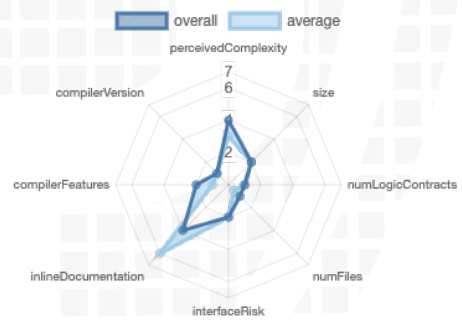| File Name	SHA-1 Hash
contracts/FeeDistributorV3.sol	4f48be169922c7aae0dfc260517d85f5ddd45f8f
contracts/InvaderPool.sol	93a94db99e663fd264d5e0d433965081fca5ef17

Metrics

Source Lines

v1.0

v1.2

Risk Level

v1.0

v1.2

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	0	2	0
1.2	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.

Version	Public	Payable
1.0	36	0
1.2	18	2

Version	External	Internal	Private	Pure	View
1.0	23	51	0	4	4
1.2	17	31	0	1	4

State Variables

Version	Total	Public
1.0	47	43
1.2	28	28

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.12` `^0.8.0`				
1.2	`0.8.12`		yes		

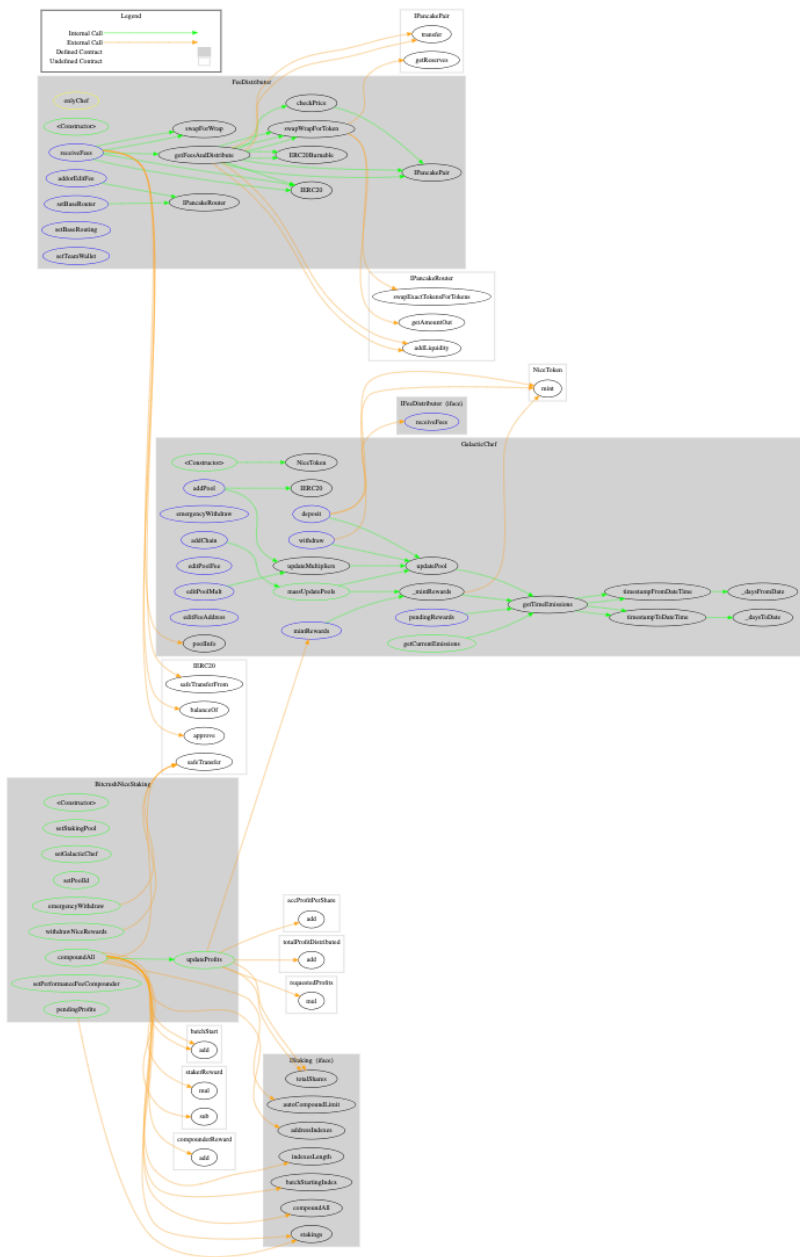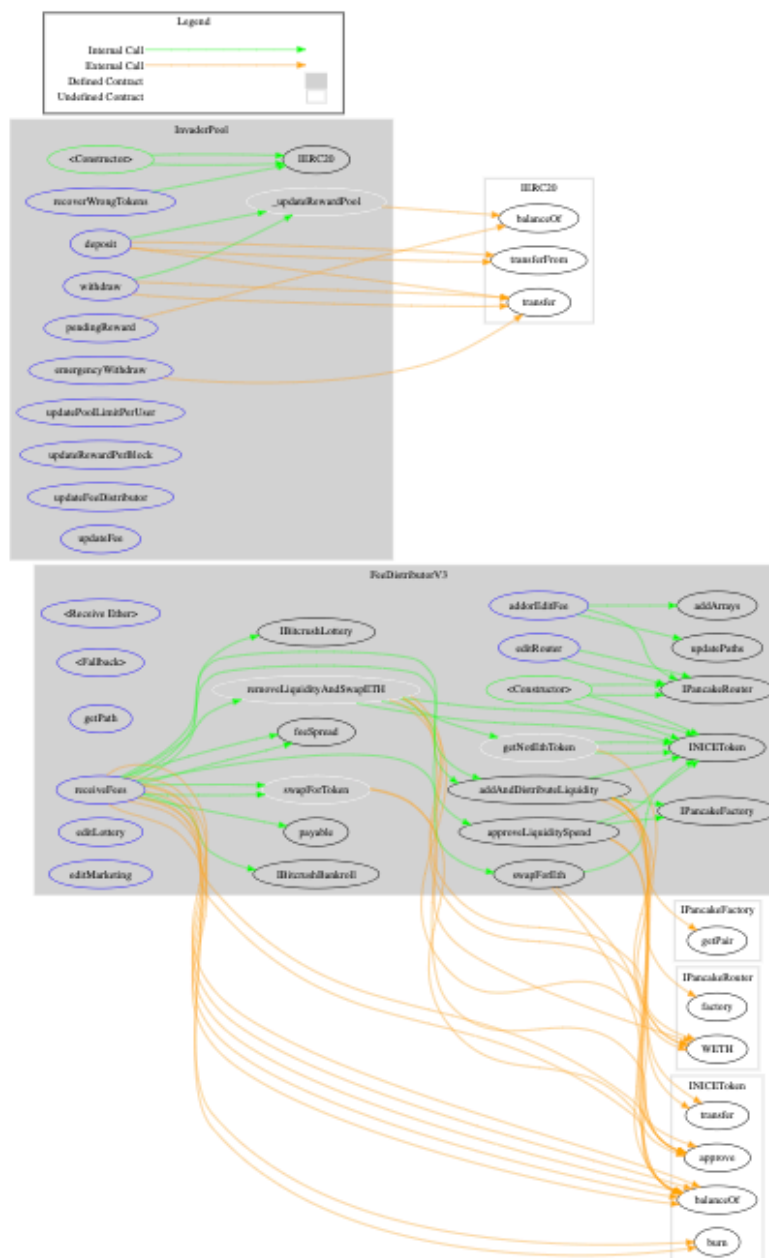| Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes					

Inheritance Graph
v1.0

v1.2

CallGraph
v1.0

v1.2
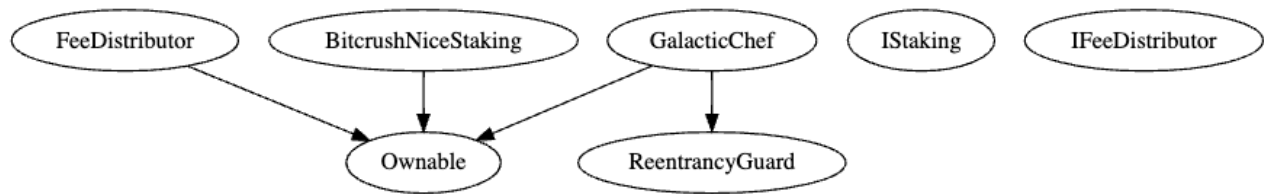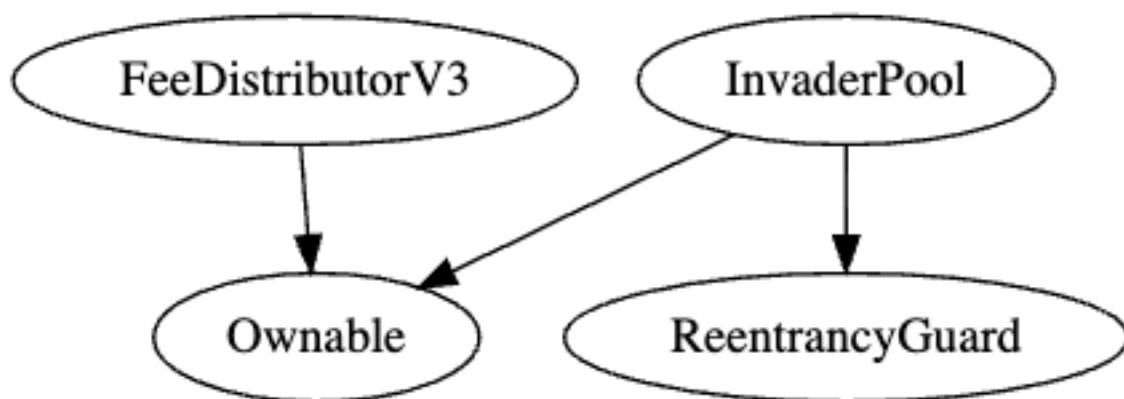
Scope of Work/Verify Claims

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

Modifiers and public functions
v1.0

Column 1:
- ˅ 💧 addPool
  - ◎ onlyOwner
- 💧 updatePool
- 💧 massUpdatePools
- ˅ 💧 mintRewards
  - ◎ nonReentrant
- ˅ 💧 deposit
  - ◎ nonReentrant
- ˅ 💧 withdraw
  - ◎ nonReentrant
- ˅ 💧 emergencyWithdraw
  - ◎ nonReentrant
- ˅ 💧 editPoolMult
  - ◎ onlyOwner
- ˅ 💧 editPoolFee
  - ◎ onlyOwner
- ˅ 💧 addChain
  - ◎ onlyOwner
- ˅ 💧 editFeeAddress
  - ◎ onlyOwner

Column 2 (top):
- ˅ 💧 renounceOwnership
  - ◎ onlyOwner
- ˅ 💧 transferOwnership
  - ◎ onlyOwner

Column 2 (bottom):
- ˅ 💧 addorEditFee
  - ◎ onlyOwner
- ˅ 💧 receiveFees
  - ◎ onlyChef
- ˅ 💧 setBaseRouter
  - ◎ onlyOwner
- ˅ 💧 setBaseRouting
  - ◎ onlyOwner
- ˅ 💧 setTeamWallet
  - ◎ onlyOwner

Column 3:
- ˅ 💧 setStakingPool
  - ◎ onlyOwner
- ˅ 💧 setGalacticChef
  - ◎ onlyOwner
- ˅ 💧 setPoolId
  - ◎ onlyOwner
- 💧 updateProfits
- 💧 pendingProfits
- 💧 compoundAll
- 💧 withdrawNiceRewards
- ˅ 💧 setPerformanceFeeCompounder
  - ◎ onlyOwner
- 💧 emergencyWithdraw

Comments
- Deployer can set following state variables without any limitations
  - niceStaking
    - poolId
- Deployer can set following addresses
  - FeeDistribution
    - tokenRouter
    - niceLiquidity
    - nicePath
    - crushLiquidity
    - crushPath
    - teamWallet

- niceStaking
  - stakingPool
  - galacticChef

- GalacticChef
  - Everybody can call updatePool function
  - OnlyOwner can increment chains state variable by calling addChain function L355

v1.2
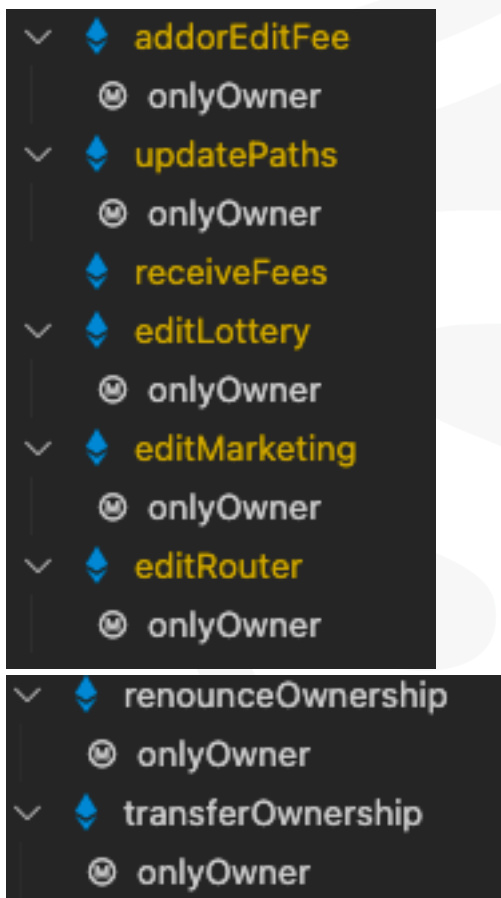
Comments
- Deployer can set following state variables without any limitations
  - InvaderPool
    - poolLimit
    - rewardPerBlock
      - As long as pool not started

- Deployer can set following addresses
  - FeeDistribution
    - tokenPath

- token0Path
- token1Path
- routerCrush
- routerNice
- marketingWallet
- lottery
  - InvaderPool
    - feeAddress

- Owner can create feeData

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/GalacticChef.sol	1	—	449	435	287	124	185	♻️
🔍	contracts/IStaking.sol	—	1	29	5	3	1	15	—
🔍	contracts/IFeeDistributor.sol	—	1	6	5	3	1	3	—
📝	contracts/FeeDistributor.sol	1	—	322	295	239	48	127	📥
📝	contracts/niceStaking.sol	1	—	177	177	126	31	90	—
📝🔍	Totals	3	2	983	917	658	205	420	📥♻️

v1.2

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/FeeDistributorV3.sol	1	—	594	529	422	104	237	💰📥
📝	contracts/InvaderPool.sol	1	—	241	234	179	27	98	📥
📝	Totals	2	—	835	763	601	131	335	💰📥

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

<div style="background-color:#7ee84e; text-align:center;">

AUDIT PASSED

</div>

Critical issues

<div style="background-color:#7ee84e; text-align:center;">

No critical issues

</div>

High issues

<div style="background-color:#7ee84e; text-align:center;">

No high issues

</div>

Medium issues

<div style="background-color:#7ee84e; text-align:center;">

No medium issues

</div>

Low issues

Issue	File	Type	Line	Description
#1	FeeDistributorV3	Missing Zero Address Validation (missing-zero-check)	577, 571, 583, 73-76	Check that the address is not zero
#2	Invader Pool	Missing Zero Address Validation (missing-zero-check)	48	Check that the address is not zero

Informational issues

Issue	File	Type	Line	Description
#1	FeeDistributorV3	Misspelling	See description	Change following words: - approviing L341 - throught L128 Make sure to change it everywhere else as well.

#2	FeeDistributorV3	Error message is missing	101, 354, 513, 572, 578, 585, 589	Provide an error message for require statement
#3	Invader Pool	Error message is missing	231	Provide an error message for require statement
#4	FeeDistributorV3	Missing return	L170	Else condition returns value but If condition doesn't

#5	Invader Pool	Require statement check + wrong comment in L210	212	Following require statement can be bypassed:
				
				require(
				    _newLimit > poolLimit || _newLimit == 0,
				    "Increase or remove limit only"
				    );
				_newLimit must be 0 or higher than previousPoolLimit but it can set lower than previous poolLimit by setting it first to 0 and then to a lower poolLimit as before

Audit Comments
01. April 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	**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