SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

InpulseX Staking

Audit

Security Assessment
09. June, 2023

For

IMPULSEX

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	23. February 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	9. March 2023	• Reaudit
1.2	9. June 2023	• Added New Files to the Audit Scope

Network
Ethereum, BSC, Avalanche, and Polygon

Website
http://www.inpulsex.io/

Telegram
https://t.me/InpulseX_Official

Twitter
https://twitter.com/InpulseX_io

Discord
https://discord.gg/kH6PaHsNHK

Facebook
https://www.facebook.com/InpulseX/

Instagram
http://www.instagram.com/the_nftx/

TikTok
https://www.tiktok.com/@inpulsex_official

Medium
https://medium.com/@InpulseX_Official

Description

InpulseX is an ambitious project created to offer unwavering support to the biggest mission of humankind, which is to become a multiplanetary species.
The InpulseX ecosystem will take the lead within the blockchain community, bringing awareness and raising financial resources to help write this exciting new chapter.
Together we will make history.

Project Engagement

During the Date of 23 February 2023, **InpulseX 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/KenshiTech/InpulseX/tree/master/staking
- Commit: b82c25db733303f34aae4363d17f608717f275ec

v1.1

- https://github.com/KenshiTech/InpulseX/tree/master/staking
- Commit: 8c872789d3d06a74ede9d7d2081a42d469be6102

v1.2

- https://github.com/KenshiTech/InpulseX/tree/master/staking
- Commit: 9922053d171c05bb30a7b22b803d35d7cf6270f3

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.

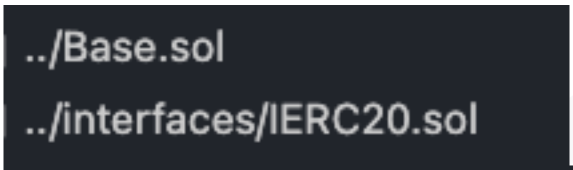
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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

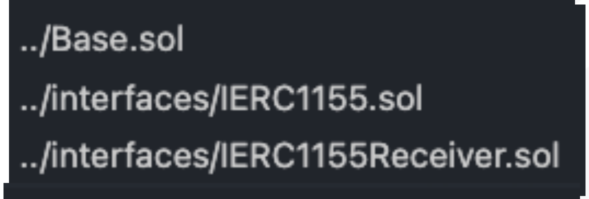
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

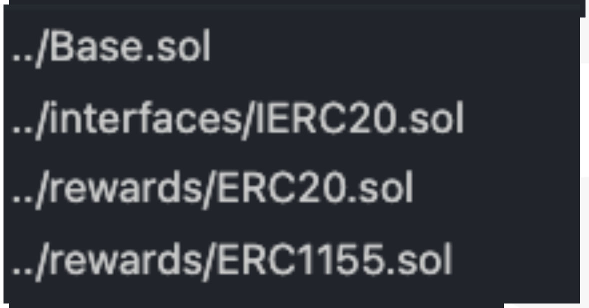
Imported packages:
v1.0



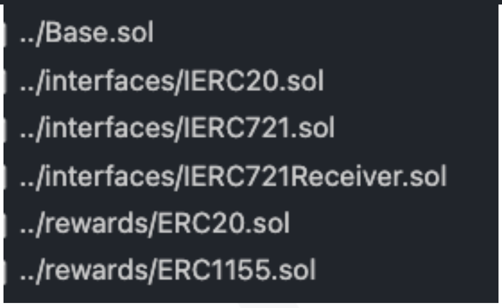
```
../Base.sol
../interfaces/IERC20.sol
```



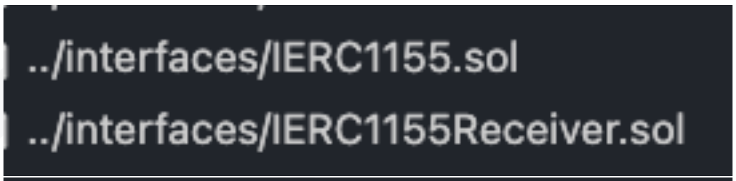
```
../Base.sol
../interfaces/IERC1155.sol
../interfaces/IERC1155Receiver.sol
```



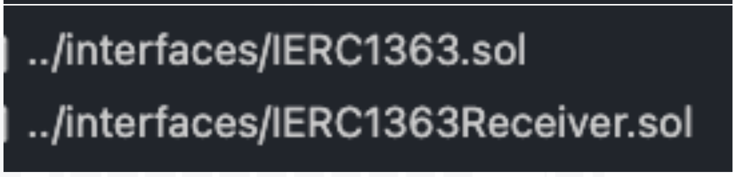
```
../Base.sol
../interfaces/IERC20.sol
../rewards/ERC20.sol
../rewards/ERC1155.sol
```



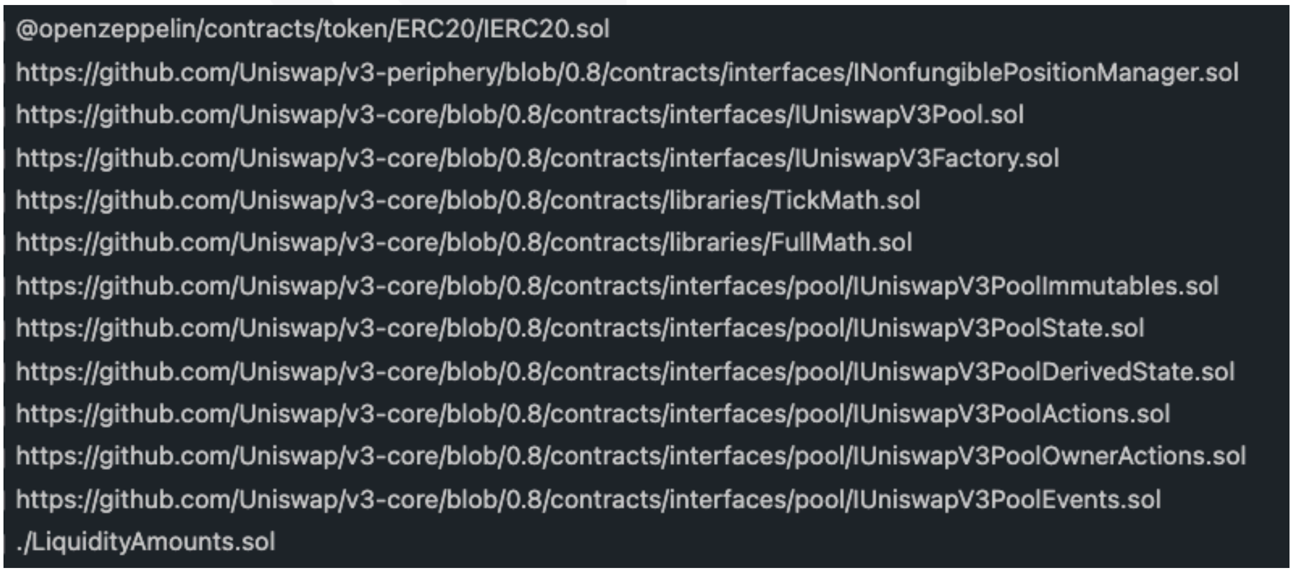
```
../Base.sol
../interfaces/IERC20.sol
../interfaces/IERC721.sol
../interfaces/IERC721Receiver.sol
../rewards/ERC20.sol
../rewards/ERC1155.sol
```



```
../interfaces/IERC1155.sol
../interfaces/IERC1155Receiver.sol
```



```
../interfaces/IERC1363.sol
../interfaces/IERC1363Receiver.sol
```



```
@openzeppelin/contracts/token/ERC20/IERC20.sol
https://github.com/Uniswap/v3-periphery/blob/0.8/contracts/interfaces/INonfungiblePositionManager.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/IUniswapV3Pool.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/IUniswapV3Factory.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/libraries/TickMath.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/libraries/FullMath.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolImmutables.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolState.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolDerivedState.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolActions.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolOwnerActions.sol
https://github.com/Uniswap/v3-core/blob/0.8/contracts/interfaces/pool/IUniswapV3PoolEvents.sol
./LiquidityAmounts.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

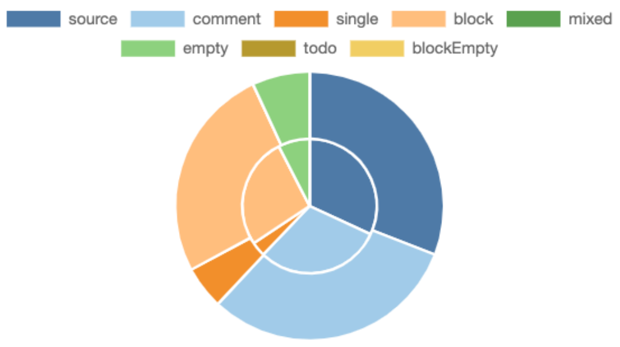
File Name	SHA-1 Hash
contracts/Repo.sol	d93e1568a2da17a604e2d63fff98d8ea18f15c7a
contracts/staking/ERC1363.sol	4e5ac98d3712da31d93dad3f2132cacacd9d7b0a
contracts/staking/UniSwapNFT.sol	e33354037f8001d0416c2a0cc9f3926d2b16cc3e
contracts/staking/ERC721.sol	f9bfcadd6d31ec9065922fe6c007f9d90475be58
contracts/staking/ERC20.sol	6413e5a97352c44b31447e5ca548c80143f1b404
contracts/staking/ERC1155.sol	ec36a32f9a9854b5a3244c7a623d00eba4c7e82a
contracts/Base.sol	8169d356f3054c384c0c6beb090341be6960f073
contracts/utils/UniSwapNFTPrice.sol	a59ea411ba83558cc7a59a8047cfe6e43152485b
contracts/utils/NFTSweep.sol	0ee0145f1c3a8e3312df2c779345f6dbde8ae8c6
contracts/utils/LiquidityAmounts.sol	e4516b9dba23922a4d25b831c0716f5bf36fd0b9
contracts/Dummy.sol	68c7f84c6180e5d44a1d1b033afd7f631ac5c9c8

contracts/rewards/ERC20.sol	60c3977198b297e0aca7d151c70ed014a9f621ae
contracts/rewards/ERC1155.sol	ca29449ea3ac3fd5845e8412a39344760ed73c54

Metrics

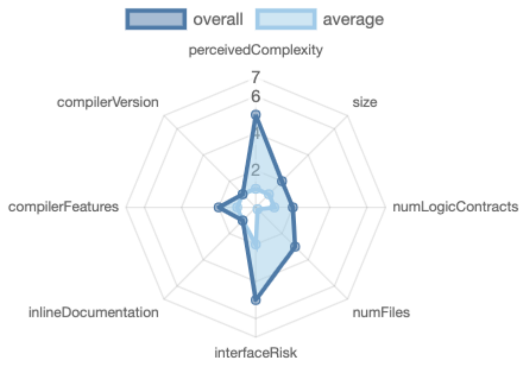
Source Lines

v1.2



Risk Level

v1.2



Capabilities
Components
v1.2

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
16	1	1	9

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
62	0

External	Internal	Private	Pure	View
44	50	1	17	25

StateVariables

Total	🌐Public
24	3

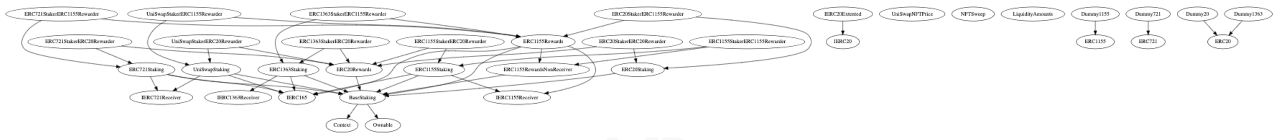
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`0.8.17`				

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

♻️ TryCatch	Σ Unchecked
`yes`	`yes`

Inheritance Graph
v1.2



CallGraph
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).

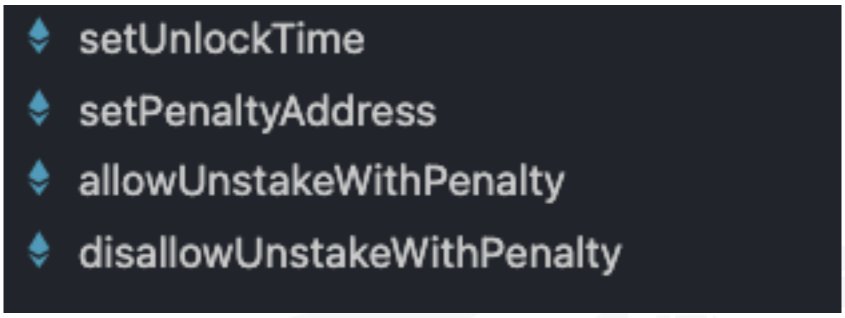
We will verify the following claims:

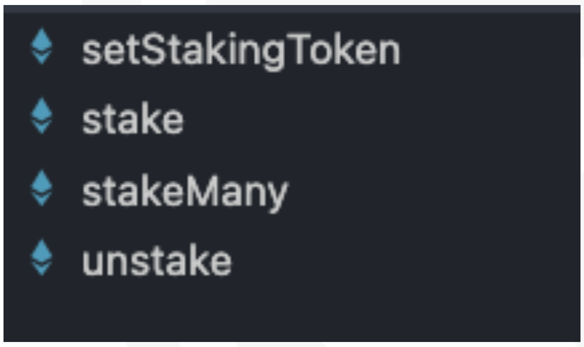
1. Is contract an upgradeable
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Deployer cannot set fees
6. Deployer cannot blacklist/antisnipe addresses
7. Overall checkup (Smart Contract Security)

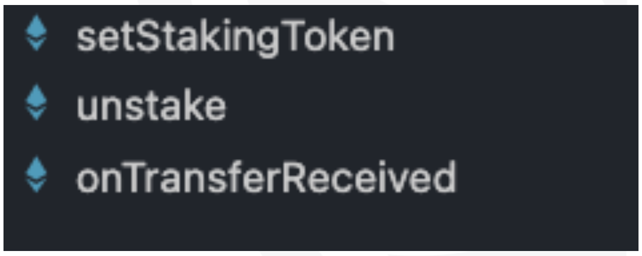
Is contract an upgradeable

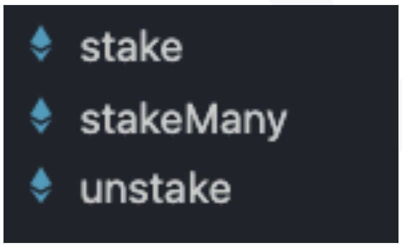
Name	
Is contract an upgradeable?	No

Write functions of contract v1.2


setUnlockTime
setPenaltyAddress
allowUnstakeWithPenalty
disallowUnstakeWithPenalty


setStakingToken
stake
stakeMany
unstake


setStakingToken
unstake
onTransferReceived


stake
stakeMany
unstake

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	–	–	–

Comments:

v1.2

- Owner cannot lock user funds by changing the staking token address because it can only be set once

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can pause	–	–	–

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.2

- The owner can set the penalty fees for any address to up to 25% only

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

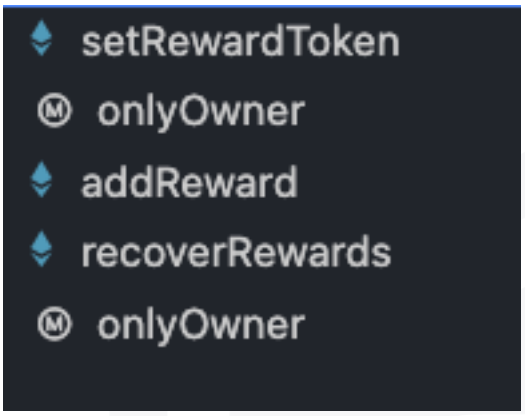
Tested	Verified
✓	✓

Legend

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

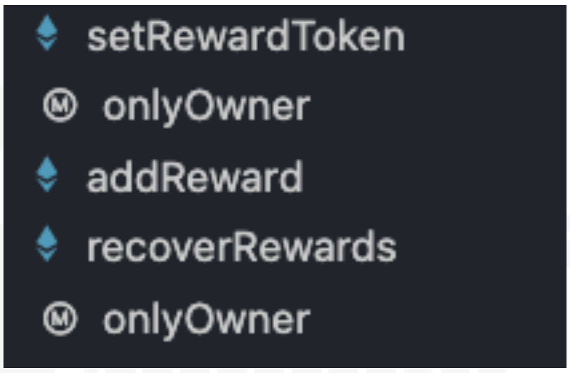
Modifiers and public functions
v1.2

rewards/ERC20.sol



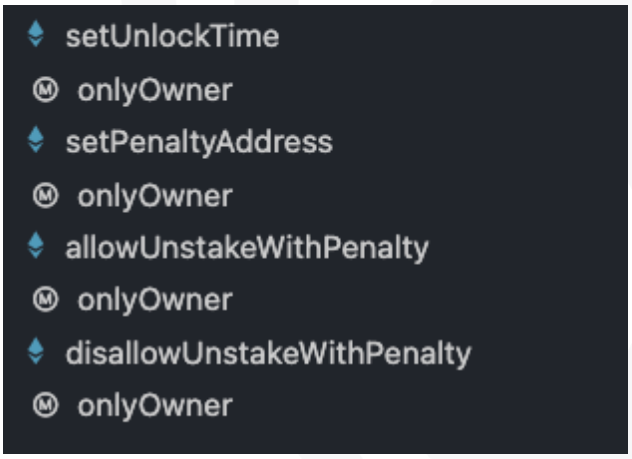
setRewardToken
onlyOwner
addReward
recoverRewards
onlyOwner

rewards/ERC1155.sol



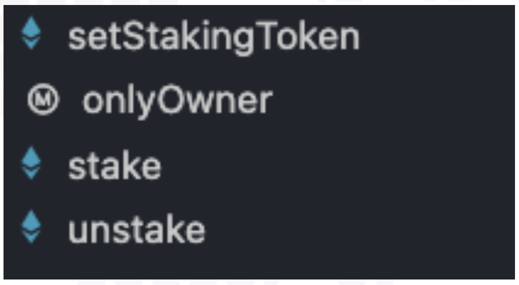
setRewardToken
onlyOwner
addReward
recoverRewards
onlyOwner

Base.sol



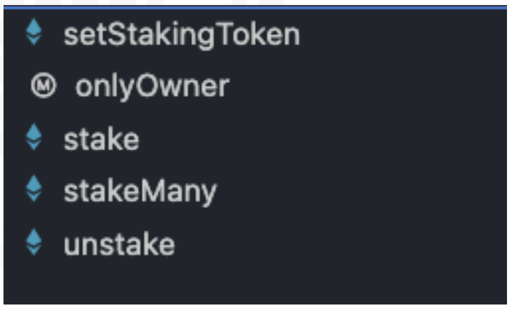
setUnlockTime
onlyOwner
setPenaltyAddress
onlyOwner
allowUnstakeWithPenalty
onlyOwner
disallowUnstakeWithPenalty
onlyOwner

Staking/ERC20.sol



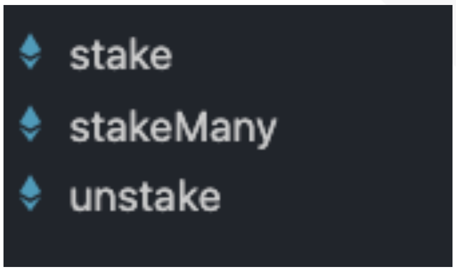
setStakingToken
onlyOwner
stake
unstake

Staking/ERC721.sol



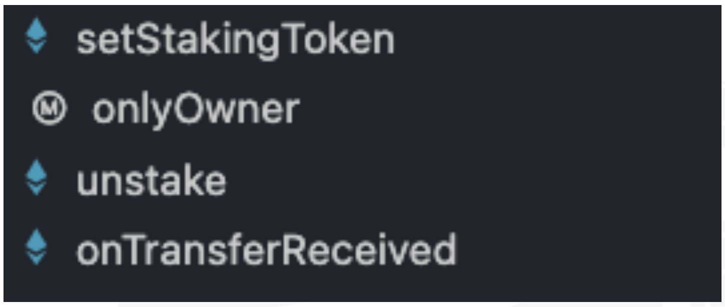
setStakingToken
onlyOwner
stake
stakeMany
unstake

UniSwapNFT.sol



stake
stakeMany
unstake

Staking/ERC1363.sol


setStakingToken
onlyOwner
unstake
onTransferReceived

Ownership Privileges:

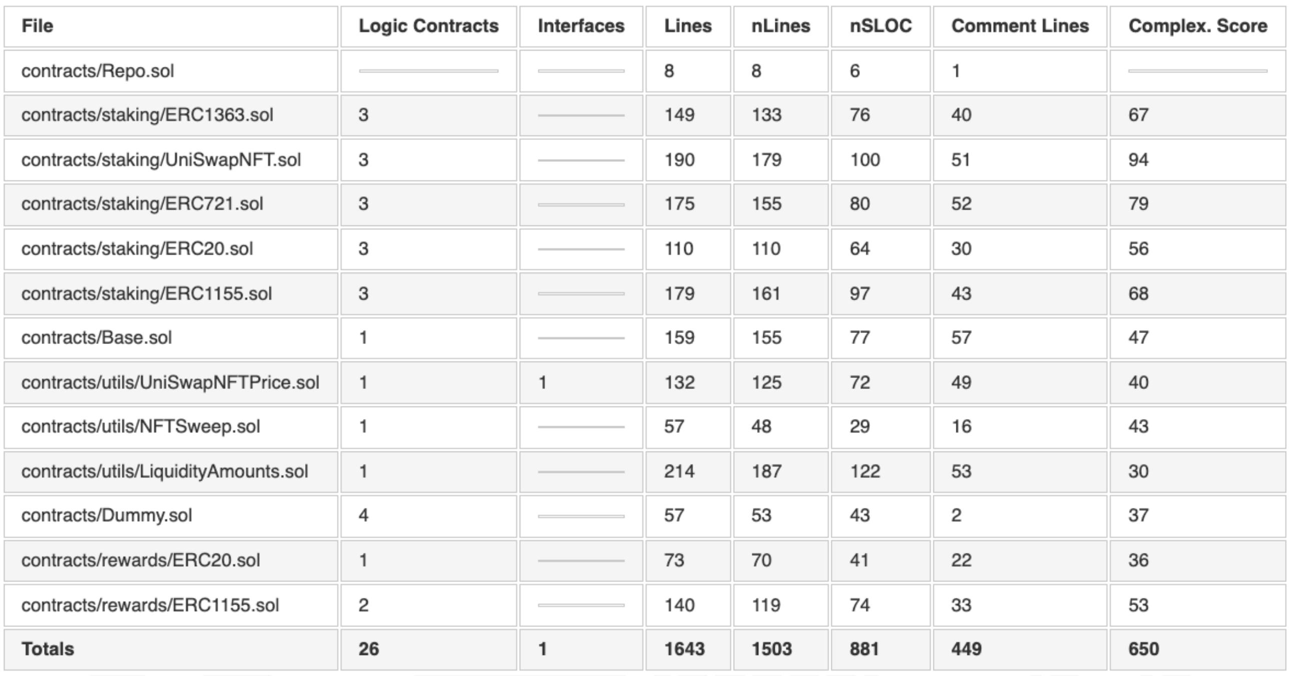
- *Base.sol:*
  - The owner can set unlock time for the staked tokens to any arbitrary value, but only once
  - Allow/Disallow users to unstake with a penalty. Therefore, the owner can do this to any address at any time, but the penalty fees cannot be more than 25%
  - Set penalty receiver address.

- *staking/ERC20.sol:*
  - The owner can update the staking token address only once, and it cannot be updated
  - **Note:** This same exists with the staking of ERC1155, ERC721, and ERC1363

- *rewards/ERC20.sol:*
  - Set/Update reward token, but only once
  - Recover the tokens from the contract. Hence, withdraw the reward token from the contract.
  - In the contract, any user can transfer the reward token, but only the owner can withdraw it.
  - **Note:** This same exists with rewards/ERC721.sol,

Source Units in Scope

v1.2



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/Repo.sol			8	8	6	1	
contracts/staking/ERC1363.sol	3		149	133	76	40	67
contracts/staking/UniSwapNFT.sol	3		190	179	100	51	94
contracts/staking/ERC721.sol	3		175	155	80	52	79
contracts/staking/ERC20.sol	3		110	110	64	30	56
contracts/staking/ERC1155.sol	3		179	161	97	43	68
contracts/Base.sol	1		159	155	77	57	47
contracts/utils/UniSwapNFTPrice.sol	1	1	132	125	72	49	40
contracts/utils/NFTSweep.sol	1		57	48	29	16	43
contracts/utils/LiquidityAmounts.sol	1		214	187	122	53	30
contracts/Dummy.sol	4		57	53	43	2	37
contracts/rewards/ERC20.sol	1		73	70	41	22	36
contracts/rewards/ERC1155.sol	2		140	119	74	33	53
Totals	**26**	**1**	**1643**	**1503**	**881**	**449**	**650**

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

No low issues

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 variables, functions etc. do.

09. June 2023:

- There is still an owner (Owner still has not renounced ownership)
- 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	**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