



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

vEmpire

Audit

Security Assessment
20. March, 2022

For



vEMPIRE

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	18
Source Units in Scope	20
Critical issues	21
High issues	21
Medium issues	21
Low issues	21
Informational issues	21
Audit Comments	32
SWC Attacks	33

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	20. March 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Ethereum (ERC20)

Website

<https://v-empire.io/>

Telegram

<https://t.me/vempirediscussion>

Twitter

<http://twitter.com/vempiredigital>

Medium

<https://medium.com/@v-empire.digital>

Discord

<https://discord.gg/Wk3aF3PNKM>

Youtube

<https://youtube.com/c/vEmpireDDAO>

Description

vEmpire DDAO is the world's largest Decentralized Metaverse Investment Organization. The official vEmpire protocol incorporates different strategies to incentivize Metaverse token staking to fund the battle against centralisation.

vEmpire is entirely focused on protecting decentralized technologies through virtual property and Metaversal asset acquisition.

Project Engagement

During the 16th of March 2022, **vEmpire 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/v-Empire/vEmpire/tree/b57ad1c065431751ceb3c2c9f7d47647293592c5>
 - Commit: b57ad1c065431751ceb3c2c9f7d47647293592c5

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 as possible.
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-upgradeable/access/AccessControlUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/cryptography/MerkleProofUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	1

Tested Contract Files

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

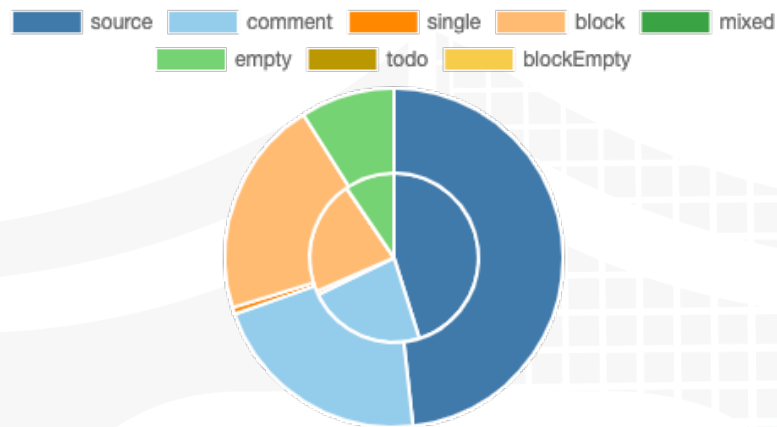
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

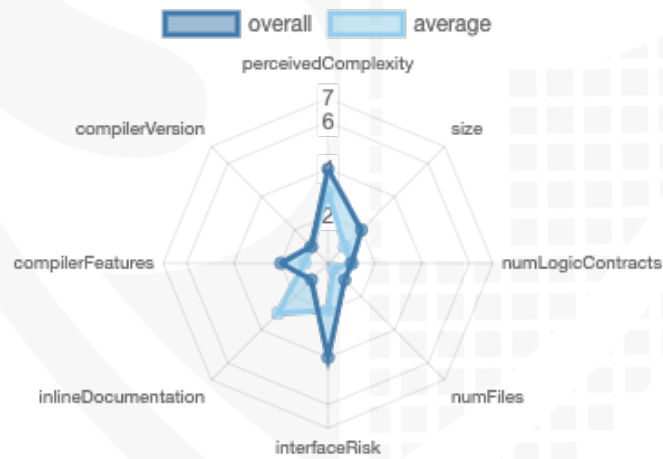
File Name	SHA-1 Hash
contracts/Staking.sol	cee40419664a8fb1bedb6cfa50dbaae80f004ec1
contracts/NFT.sol	c4e176032c3954ad9fcf83972802e0019cb3eb50
contracts/Sale.sol	145a29f7ef4dfcb2916fa691a7d471432a556b89

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	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	2

Version	External	Internal	Private	Pure	View
1.0	23	43	0	1	6

State Variables

Version	Total	Public
1.0	33	30

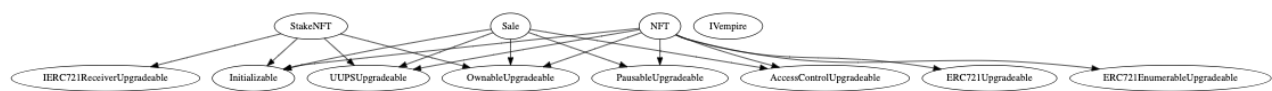
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.11		yes		

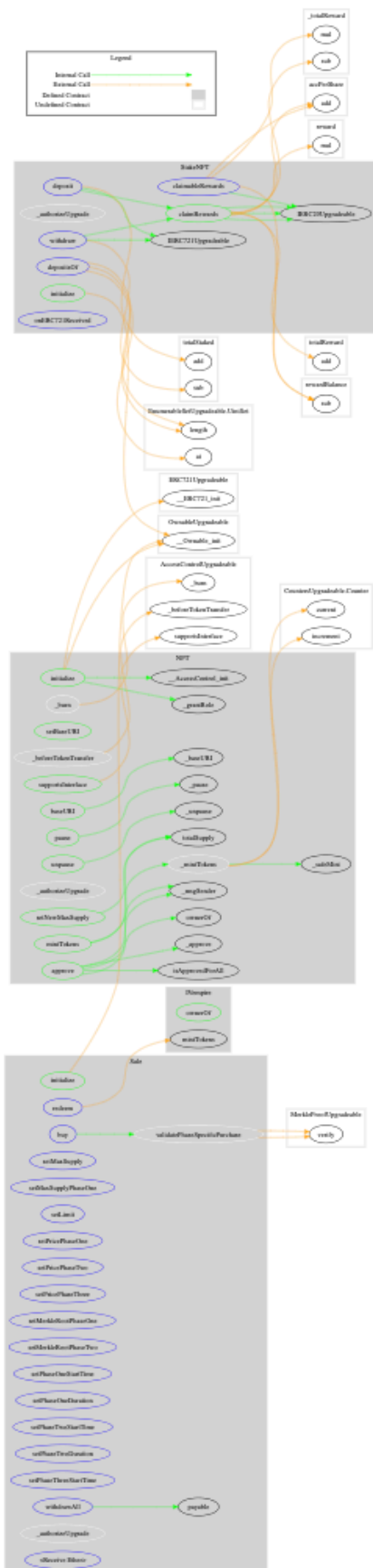
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes			yes		

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. Correct implementation of Token standard
2. Overall checkup (Smart Contract Security)



Correct implementation of Token standard

ERC721				
Function	Description	Exist	Tested	Verified
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

Write functions of contract v1.0

▼ NFT

approve

grantRole

initialize

mintTokens

pause

renounceOwnership

renounceRole

revokeRole

safeTransferFrom

safeTransferFrom

setApprovalForAll

setBaseURI

transferFrom

transferOwnership

unpause

setNewMaxSupply

upgradeTo

upgradeToAndCall

▼ SALE

buy

grantRole

initialize

redeem

renounceOwnership

renounceRole

revokeRole

setLimit

setMaxSupply

setMaxSupplyPhaseOne

setMerkleRootPhaseOne

setMerkleRootPhaseTwo

setPhaseOneDuration

setPhaseOneStartTime

setPhaseThreeStartTime

setPhaseTwoDuration

setPhaseTwoStartTime

setPricePhaseOne

setPricePhaseThree

setPricePhaseTwo

transferOwnership

upgradeTo

upgradeToAndCall

withdrawAll

▼ STAKENFT

claimRewards

deposit

initialize

renounceOwnership

transferOwnership

upgradeTo

upgradeToAndCall

withdraw



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



Information: Not listed functions are directly imported functions from library (openzeppelin)

Comments











- Deployer can set following state variables without any limitations
 - NFT
 - MAX_SUPPLY
 - Can be set only above total supply
 - Sale

- maxSupply
 - maxSupplyPhaseOne
 - limit
 - phaseOnePrice
 - phaseTwoPrice
 - phaseThreePrice
 - phaseOneStartTime
 - phaseOneDuration
 - phaseTwoStartTime
 - phaseTwoDuration
 - phaseThreeStartTime
- Deployer can enable/disable following state variables
 - _paused
 - Deployer can set following addresses/urls
 - _baseURIValue
 - mintTokens can only be called from minter_role, if contract is not paused
 - Owner can set
 - merkleRootPhaseOne
 - merkleRootPhaseTwo

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/Staking.sol	1	————	235	219	143	49	120	
	contracts/NFT.sol	1	————	197	171	94	53	80	
	contracts/Sale.sol	2	————	416	398	223	129	145	 
	Totals	4	————	848	788	460	231	345	 

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

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Staking	Missing Zero Address Validation (missing-zero-check)	58	Check that the address is not zero
#2	NFT	Local variables shadowing	122, 123, 126, 52, 53, 56	Rename the local variables that shadow another component

Informational issues

Issue	File	Type	Line	Description
#1	NFT	Functions that are not used	168	Remove unused functions
#2	Staking	Unused state variables	37	Remove unused state variables

#3	Staking	Misspelling		Change following words: - deposte to deposited L26
	NFT	Misspelling		Change following words: - minitng to minting L88 - miniting to minting L95

Test Protocol

Contract: ERC721

Contract interface

ERC165

ERC165's supportsInterface(bytes4)

- ✓ uses less than 30k gas [skip-on-coverage]
- ✓ claims support

supportsInterface(bytes4)

- ✓ has to be implemented

ERC721

ERC165's supportsInterface(bytes4)

- ✓ uses less than 30k gas [skip-on-coverage]
- ✓ claims support

balanceOf(address)

- ✓ has to be implemented

ownerOf(uint256)

- ✓ has to be implemented

approve(address,uint256)

- ✓ has to be implemented

getApproved(uint256)

- ✓ has to be implemented

setApprovalForAll(address,bool)

- ✓ has to be implemented

isApprovedForAll(address,address)

- ✓ has to be implemented

transferFrom(address,address,uint256)

- ✓ has to be implemented

safeTransferFrom(address,address,uint256)

- ✓ has to be implemented

safeTransferFrom(address,address,uint256,bytes)

- ✓ has to be implemented

with minted tokens

balanceOf

when the given address owns some tokens

- ✓ returns the amount of tokens owned by the given address

when the given address does not own any tokens

- ✓ returns 0

when querying the zero address

- ✓ throws

ownerOf

when the given token ID was tracked by this token

- ✓ returns the owner of the given token ID

when the given token ID was not tracked by this token

- ✓ reverts

transfers

via transferFrom

when called by the owner

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event

(node:47847) DeprecationWarning: expectEvent.inLogs() is deprecated.
Use expectEvent() instead.

- ✓ clears the approval for the token ID

- ✓ emits an Approval event

- ✓ adjusts owners balances

- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event

- ✓ clears the approval for the token ID

- ✓ emits an Approval event

- ✓ adjusts owners balances

- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event

- ✓ clears the approval for the token ID

- ✓ emits an Approval event

- ✓ adjusts owners balances

- ✓ adjusts owners tokens by index

when called by the owner without an approved user

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event

- ✓ clears the approval for the token ID

- ✓ emits an Approval event

- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts

when the address to transfer the token to is the zero address

- ✓ reverts

via safeTransferFrom

with data

to a user account

when called by the owner

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the owner without an approved user

✓ transfers the ownership of the given token ID to the given address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts

when the address to transfer the token to is the zero address

- ✓ reverts

to a valid receiver contract

- ✓ calls onERC721Received
- ✓ calls onERC721Received from approved

when called by the owner

- ✓ transfers the ownership of the given token ID to the given address

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given address

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given address

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID

- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the owner without an approved user

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts

when the address to transfer the token to is the zero address

- ✓ reverts

with an invalid token id

- ✓ reverts

without data

to a user account

when called by the owner

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the owner without an approved user

- ✓ transfers the ownership of the given token ID to the given

address (60ms)

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts

when the address to transfer the token to is the zero address

- ✓ reverts

to a valid receiver contract

- ✓ calls onERC721Received
- ✓ calls onERC721Received from approved

when called by the owner

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event

- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the owner without an approved user

- ✓ transfers the ownership of the given token ID to the given

address

- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ emits an Approval event
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts

when the address to transfer the token to is the zero address

- ✓ reverts

with an invalid token id

- ✓ reverts

to a receiver contract returning unexpected value

- ✓ reverts

to a receiver contract that reverts with message

- ✓ reverts

to a receiver contract that reverts without message

- ✓ reverts

to a receiver contract that panics

- ✓ reverts

to a contract that does not implement the required function

- ✓ reverts

approve

- when clearing approval
 - when there was no prior approval
 - ✓ clears approval for the token
 - ✓ emits an approval event
 - when there was a prior approval
 - ✓ clears approval for the token
 - ✓ emits an approval event
- when approving a non-zero address
 - when there was no prior approval
 - ✓ sets the approval for the target address
 - ✓ emits an approval event
 - when there was a prior approval to the same address
 - ✓ sets the approval for the target address
 - ✓ emits an approval event
 - when there was a prior approval to a different address
 - ✓ sets the approval for the target address
 - ✓ emits an approval event
- when the address that receives the approval is the owner
 - ✓ reverts
- when the sender does not own the given token ID
 - ✓ reverts
- when the sender is approved for the given token ID
 - ✓ reverts
- when the sender is an operator
 - ✓ sets the approval for the target address
 - ✓ emits an approval event
- when the given token ID does not exist
 - ✓ reverts

setApprovalForAll

- when the operator willing to approve is not the owner
 - when there is no operator approval set by the sender
 - ✓ approves the operator
 - ✓ emits an approval event
 - when the operator was set as not approved
 - ✓ approves the operator
 - ✓ emits an approval event
 - ✓ can unset the operator approval
 - when the operator was already approved
 - ✓ keeps the approval to the given address
 - ✓ emits an approval event
- when the operator is the owner
 - ✓ reverts

getApproved

when token is not minted

- ✓ reverts

when token has been minted

- ✓ should return the zero address

when account has been approved

- ✓ returns approved account

Sale contract

Initial configuration

- ✓ Should set the right owner NFT
- ✓ Should set the right owner of sale
- ✓ Total supply of NFT should be 0

Check owner condition

- ✓ Should non owner tries to call setMaxSupply
- ✓ Should non owner tries to call setMaxSupplyPhaseOne
- ✓ Should non owner tries to call setLimit
- ✓ Should non owner tries to call setPricePhaseOne
- ✓ Should non owner tries to call setPricePhaseTwo
- ✓ Should non owner tries to call setPricePhaseThree
- ✓ Should non owner tries to call setPhaseOneStartTime
- ✓ Should non owner tries to call setPhaseOneDuration
- ✓ Should non owner tries to call setPhaseTwoStartTime
- ✓ Should non owner tries to call setPhaseTwoDuration
- ✓ Should non owner tries to call setPhaseThreeStartTime
- ✓ Should non owner tries to call setMerkleRootPhaseOne
- ✓ Should non owner tries to call setMerkleRootPhaseTwo

Phase one minting

- ✓ Buy single NFT
- ✓ Buy Multiple NFTs
- ✓ Claim with single NFTs
- ✓ Claim with multiple NFTs (40ms)
- ✓ Set new purchase amount
- ✓ Should revert if user tries to claim more than allotted
- ✓ Should revert if user tries to buy more than allotted (45ms)
- ✓ Should revert if user tries to buy 0 tokens
- ✓ Should revert if user tries to buy after sale time
- ✓ Should revert if user tries to buy before sale time
- ✓ Should revert if user tries to buy after Phase max supply reached
- ✓ Should revert if another user tries to buy
- ✓ Should revert if user tries to buy with less amount
- ✓ Should revert if user tries to buy multiple tokens with less amount

Phase Two minting

- ✓ Buy single NFT
- ✓ Buy Multiple NFTs
- ✓ Claim with single NFTs

- ✓ Claim with multiple NFTs (39ms)
- ✓ Set new purchase amount
- ✓ Should revert if user tries to claim more than allotted
- ✓ Should revert if user tries to buy 0 tokens
- ✓ Should revert if user tries to buy after sale time
- ✓ Should revert if user tries to buy before sale time
- ✓ Should revert if another user tries to buy
- ✓ Should revert if user tries to buy with less amount
- ✓ Should revert if user tries to buy multiple tokens with less amount

Phase Three minting

- ✓ Buy single NFT
- ✓ Buy Multiple NFTs
- ✓ Claim with single NFTs
- ✓ Claim with multiple NFTs (43ms)
- ✓ Set new purchase amount
- ✓ Should revert if user tries to claim more than allotted (38ms)
- ✓ Should revert if user tries to buy 0 tokens
- ✓ Should revert if user tries to buy before sale time
- ✓ Should revert if user tries to buy after Phase max supply reached
- ✓ Should revert if another user tries to buy
- ✓ Should revert if user tries to buy with less amount
- ✓ Should revert if user tries to buy multiple tokens with less amount

Staking contract

Initial configuration

- ✓ Should set the right owner NFT token
- ✓ Should set the right owner of stake contract
- ✓ Should check the total supply of NFT

Stake token

- ✓ Should stake the token in staking contract
- ✓ Should stake the multiple token in staking contract

Rewards

- ✓ Reward should be divided among multiple staker (94ms)
- ✓ Reward should not get the tokens from previous rewards (85ms)
- ✓ User should get the correct tokens on claim (92ms)
- ✓ User should get the correct tokens on claim when multiple users

stake (154ms)

Claim rewards after multiple stake/claim

- ✓ User should get the correct tokens on deposit again (140ms)
- ✓ User should get the correct tokens on multiple deposit/withdraw too

(320ms)

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/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.

20. March 2022:

- Read whole report for more information



SWC Attacks

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

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

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

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "SolidProof" in a white, handwritten-style script. The "P" is large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY