



DustLabs

y00ts Bridge

SMART CONTRACT AUDIT

09.10.2023

Made in Germany by Softstack.io



Table of contents

1. Disclaimer.....	3
2. About the Project and Company	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level	6
4. Auditing Strategy and Techniques Applied.....	7
4.1 Methodology	7
5. Metrics	8
5.1 Tested Contract Files	8
5.2 CallGraph.....	9
5.3 Inheritance Graph.....	10
5.4 Source Lines & Risk.....	11
5.5 Capabilities	12
5.6 Dependencies and External Imports	13
5.7 Source Unites in Scope	14
6. Scope of Work.....	16
6.1 Findings Overview	17
6.2 Manual and Automated Vulnerability Test.....	18
6.3 SWC Attacks	19
6.4 Verify Claims	23
6.5 Unit Tests.....	24
7. Executive Summary.....	28
8. About the Auditor	29



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Dust Labs. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (02.10.2023)	Layout
0.4 (04.10.2023)	Automated Security Testing Manual Security Testing
0.5 (05.10.2023)	Verify Claims and Test Deployment
0.6 (06.10.2023)	Testing SWC Checks
0.9 (06.10.2023)	Summary and Recommendation
1.0 (09.10.2023)	Final document



2. About the Project and Company

Company

DUST Labs, Inc.
Reg.: 88-3587023 8605
Santa Monica Blvd Suite 86289
West Hollywood, California
90069-4109 USA



Website:

<https://www.y00ts.com>

Twitter:

<https://twitter.com/y00tsNFT>

Discord:

<https://discord.gg/y00ts>

Instagram:

<https://instagram.com/they00ts>

2.1 Project Overview

y00ts is an avant-garde NFT project that has made waves in the digital art space. Originating on the Solana blockchain, it empowers users to create and sell unique PFPs (Profile Pictures), each distinguished by its individualistic design and metadata layers. The project is the brainchild of Dust Labs, the same visionary team that brought the acclaimed DeGods NFT collection to the forefront of the NFT world.

Central to the y00ts collection is the sheep avatar, a symbol of both simplicity and depth. With a total of 15,000 NFTs, each y00ts avatar can be minted using 375 DUST tokens, adding an element of exclusivity and value. Beyond its artistic appeal, y00ts is revolutionizing the concept of copyright and ownership in the NFT realm. Instead of centralized control, y00ts envisions a decentralized registry where intellectual property rights are transparently listed, ensuring fair and open access to all.

The project's commitment to innovation doesn't stop there. Recognizing the growing demand and the potential of multi-chain operations, this bridge aims to seamlessly connect the Solana blockchain with Ethereum and Polygon, allowing for fluid interoperability and expanding the reach of y00ts to a broader audience. This strategic move not only underscores y00ts' adaptability but also its vision to be at the forefront of blockchain technology and NFT innovation.

3. 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

4. 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.

4.1 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 softstack 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 softstack 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.



5. Metrics

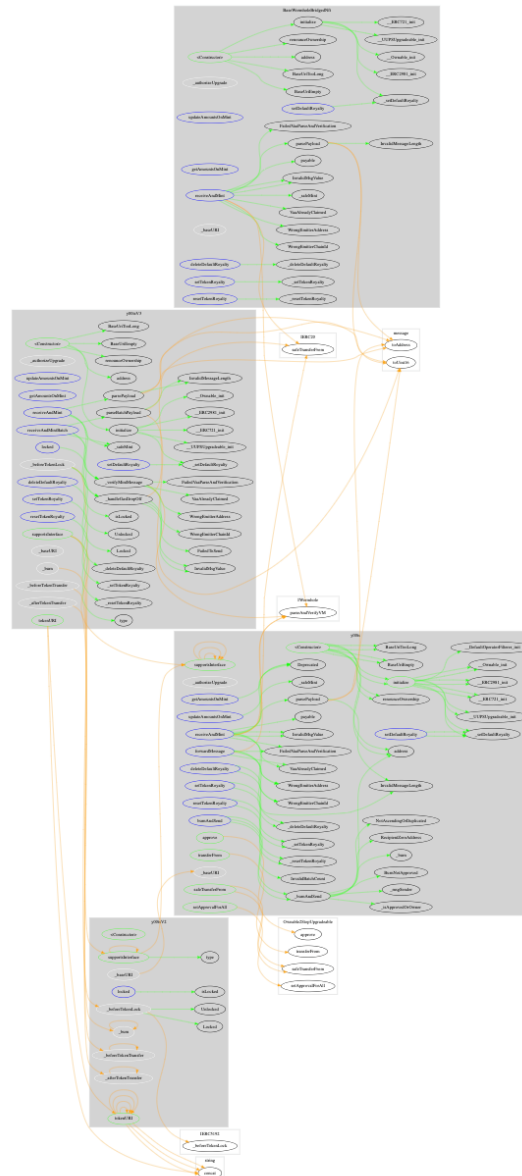
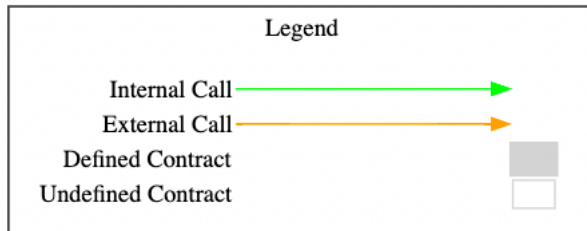
The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

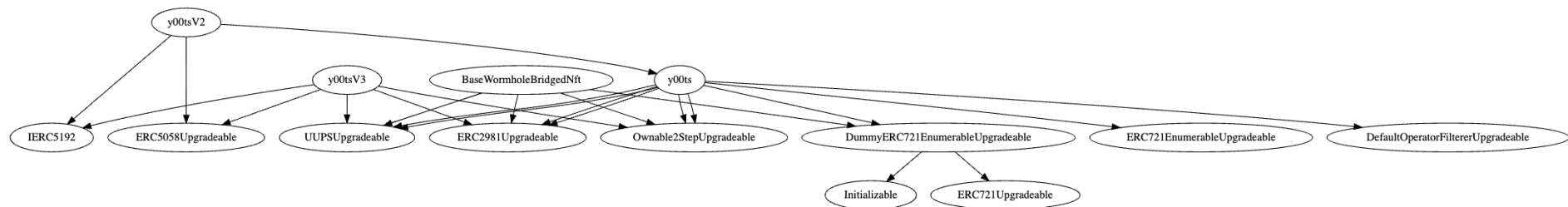
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./evm/src/nft/y00ts.sol	0016588e8f9669b970be313050720dcf
./evm/src/nft/BaseWormholeBridgedNft.sol	a11114e8a977d182b8590c159393fa52
./evm/src/nft/y00ts_royalties.sol	b2d3adfc4d573f043974d5baed1e4315
./evm/src/nft/y00tsV2.sol	88d83bda797827850c9403a8344b9e01
./evm/src/nft/y00tsV3.sol	b1cc34800e5c3b9e9a3f2f4346b4d425
./evm/src/nft/DummyERC721EnumerableUpgradeable.sol	f389f21ddba46c857ffd56cfe9b18aca

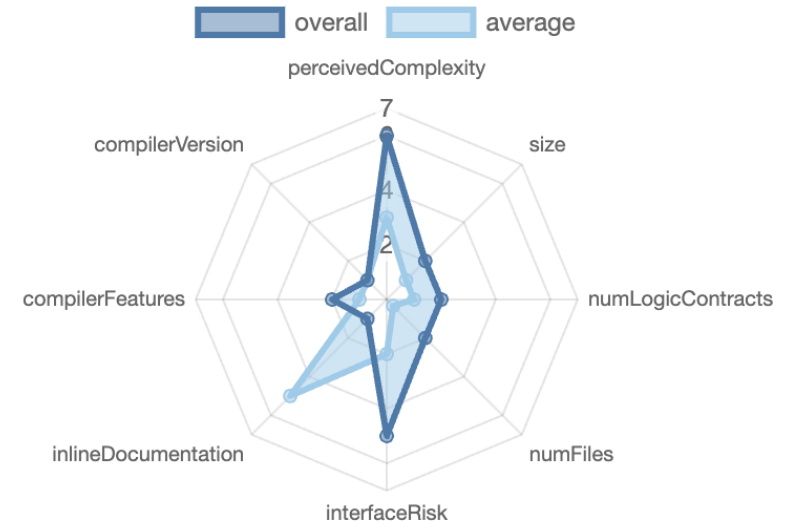
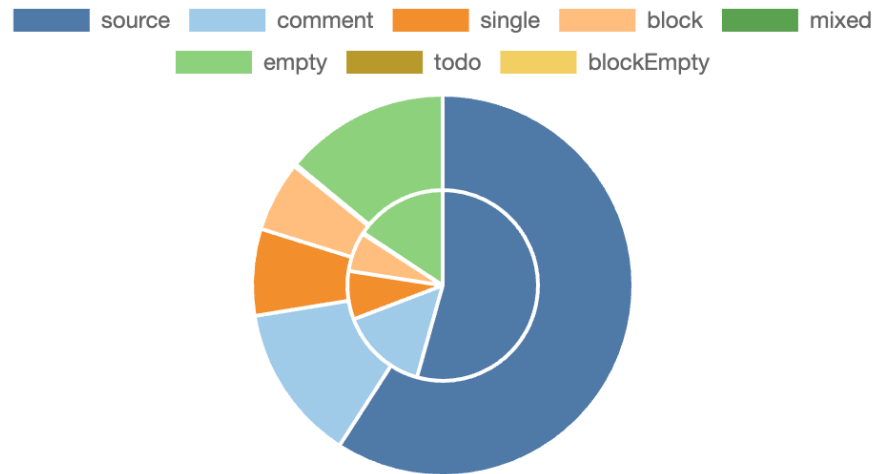
5.2 CallGraph




5.3 Inheritance Graph



5.4 Source Lines & Risk





5.5 Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
0.8.19				yes		yes (4 asm blocks)					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTRecover		 New/Create/Create2	
yes											

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				
53	8				
External	Internal	Private	Pure	View	
34	73	0	4	21	

StateVariables









Total	 Public
47	0









5.6 Dependencies and External Imports

Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/access/Ownable2StepUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/access/Ownable2StepUpgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/proxy/utils/UUPSUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/token/ERC721/ERC721Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/token/ERC721/IERC721Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/common/ERC2981Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.8.1/contracts/token/common/ERC2981Upgradeable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.8.1/contracts/token/ERC20/IERC20.sol

Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.8.1/contracts/token/ERC20/utils/SafeERC20.sol
opensea/DefaultOperatorFiltererUpgradeable.sol	https://github.com/ProjectOpenSea/operator-filter-registry/tree/main/src/DefaultOperatorFiltererUpgradeable.sol

5.7 Source Unites in Scope

Type	File	Logic Contr acts	Interface s	Lin es	nLin es	nSL OC	Comm ent Lines	Comp lex. Score	Capabiliti es
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/DummyERC721EnumerableUpgradeable.sol	1	_____	17	17	10	5	10	_____
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/y00tsV3.sol	1	_____	345	305	195	52	169	
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/y00tsV2.sol	1	_____	91	64	50	2	29	_____
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/y00ts_royalties.sol	1	_____	260	224	149	34	137	
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/BaseWormholeBridgedNft.sol	1	_____	213	191	125	29	114	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	wormhole-bridge-ethereum-migration/dust-bridging/evm/src/nft/y00ts.sol	1	_____	304	280	165	68	157	  Σ
 	Totals	6	_____	1230	1081	694	190	616	   Σ

- ☐ **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, ...)

6. Scope of Work

The DustLabs Team provided us with the files that needs to be tested. The scope of the audit are the y00ts wormhole bridge contracts.

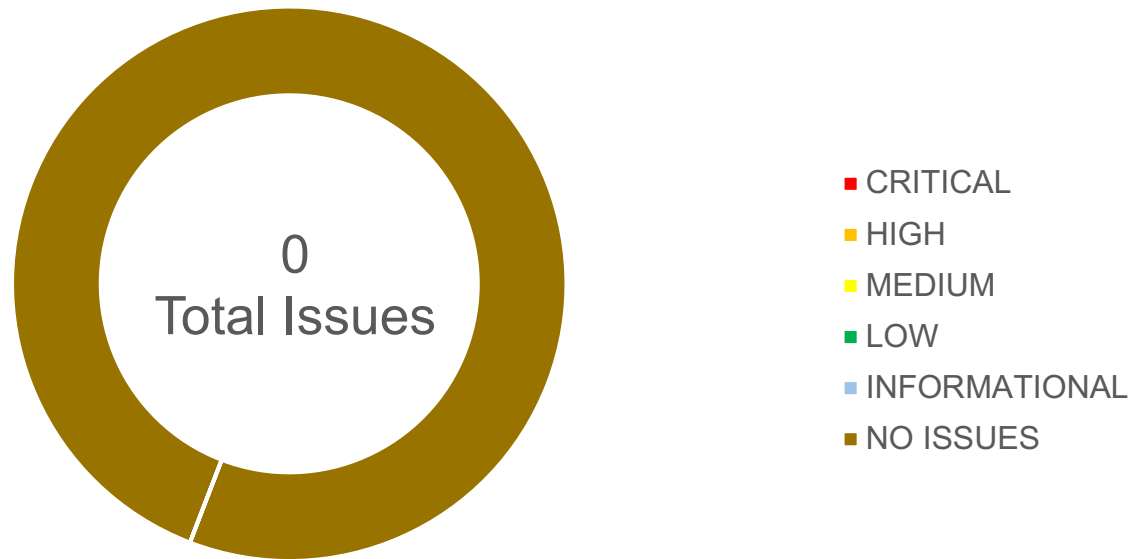
The team put forward the following assumptions regarding the security, usage of the contracts:

1. **Storage Layout Consistency:** Verify that the storage layout of ``DummyERC721EnumerableUpgradeable`` is consistent and identical with the original ``ERC721EnumerableUpgradeable`` to ensure seamless upgradability and interoperability.
2. **Access Control:** Ensure that only the rightful owners or authorized roles can interact with key functionalities, preventing unauthorized access or manipulation.
3. **Token Management:** Confirm that the mapping and management of token ownerships (``_ownedTokens``, ``_ownedTokensIndex``, ``_allTokens``, ``_allTokensIndex``) are handled correctly, without any risk of token loss or duplication.
4. **Vulnerability and Best Practices:** Perform a thorough vulnerability assessment to identify any potential weaknesses, logic flaws, or deviations from best coding practices that could expose the contract to risks.
5. **Gas Efficiency:** Check that the contract's operations are optimized for gas usage, ensuring that users won't be overcharged for executing transactions.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



6.1 Findings Overview



6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, softstack's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, softstack's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, softstack's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, softstack's experts found **no Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, softstack's experts found **no Informational issues** in the code of the smart contract.

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓


ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓


ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

6.4 Verify Claims

6.4.1 Storage Layout Consistency: Verify that the storage layout of ``DummyERC721EnumerableUpgradeable`` is consistent and identical with the original ``ERC721EnumerableUpgradeable`` to ensure seamless upgradability and interoperability.

Status: tested and verified 


6.4.2 Access Control: Ensure that only the rightful owners or authorized roles can interact with key functionalities, preventing unauthorized access or manipulation.

Status: tested and verified 


6.4.3 Token Management: Confirm that the mapping and management of token ownerships (``_ownedTokens``, ``_ownedTokensIndex``, ``_allTokens``, ``_allTokensIndex``) are handled correctly, without any risk of token loss or duplication.

Status: tested and verified 

6.4.4 Vulnerability and Best Practices: Perform a thorough vulnerability assessment to identify any potential weaknesses, logic flaws, or deviations from best coding practices that could expose the contract to risks.

Status: tested and verified 

6.4.5 Gas Efficiency: Check that the contract's operations are optimized for gas usage, ensuring that users won't be overcharged for executing transactions.

Status: tested and verified 

6.5 Unit Tests

Running 1 test for forge-test/DeBridge.t.sol:TestDeGodsBridge

[32m[PASS][0m testTokenURI() (gas: 191354)

Test result: [32mok[0m. [32m1[0m passed; [31m0[0m failed; [33m0[0m skipped; finished in 1.68s

Running 56 tests for forge-test/y00tsBridge.t.sol:TestY00tsMigration

[32m[PASS][0m testBurnAndSendBatch(uint256,uint256) (runs: 256, μ : 203167, \sim : 193241)

[32m[PASS][0m testBurnAndSendOnPolygon(uint16) (runs: 256, μ : 151150, \sim : 151152)

[32m[PASS][0m testCannotBurnAndSendBatchBurnNotApproved() (gas: 854717)

[32m[PASS][0m testCannotBurnAndSendBatchBurnNotApprovedSingleToken() (gas: 1657990)

[32m[PASS][0m testCannotBurnAndSendBatchDuplicateTokenIds() (gas: 220323)

[32m[PASS][0m testCannotBurnAndSendBatchNotAscendingTokenIds() (gas: 1028788)

[32m[PASS][0m testCannotBurnAndSendBatchOneToken() (gas: 82036)

[32m[PASS][0m testCannotBurnAndSendBatchRecipientZeroAddress() (gas: 209499)

[32m[PASS][0m testCannotBurnAndSendBatchTooManyTokens() (gas: 1035623)

[32m[PASS][0m testCannotBurnAndSendBatchZeroTokens() (gas: 24984)

[32m[PASS][0m testCannotBurnAndSendOnPolygonBurnNotApproved(uint16) (runs: 256, μ : 76923, \sim : 76923)

[32m[PASS][0m testCannotBurnAndSendOnPolygonLocked(uint16) (runs: 256, μ : 154800, \sim : 154800)

[32m[PASS][0m testCannotBurnAndSendOnPolygonRecipientZeroAddress(uint16) (runs: 256, μ : 105201, \sim : 105201)

[32m[PASS][0m testCannotForwardMessageAgain() (gas: 145246)

[32m[PASS][0m testCannotForwardMessageInvalidMessageLength() (gas: 63172)

[32m[PASS][0m testCannotForwardMessageWrongEmitterAddress() (gas: 65042)

[32m[PASS][0m testCannotForwardMessageWrongEmitterChainId() (gas: 62960)

[32m[PASS][0m testCannotGetAmountsOnMintY00tsV2Deprecated() (gas: 15002)

[32m[PASS][0m testCannotLockAutoExpoNotSupportedY00tsV2() (gas: 73713)

[32m[PASS][0m testCannotLockAutoExpoNotSupportedY00tsV3() (gas: 73575)

[32m[PASS][0m testCannotReceiveAndMintBatchAgain() (gas: 370498)

[32m[PASS][0m testCannotReceiveAndMintBatchInvalidMessageValueRelayer() (gas: 293469)

[32m[PASS][0m testCannotReceiveAndMintBatchInvalidMessageValueSelfRedeem() (gas: 298761)

[32m[PASS][0m testCannotReceiveAndMintBatchModuloNonzero() (gas: 123725)

[32m[PASS][0m testCannotReceiveAndMintBatchNotEnoughBytes() (gas: 120898)

[32m[PASS][0m testCannotReceiveAndMintBatchWrongEmitterAddress() (gas: 102076)
[32m[PASS][0m testCannotReceiveAndMintBatchWrongEmitterChainId() (gas: 101052)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumAgain() (gas: 245130)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumInvalidMessageLength() (gas: 84964)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumInvalidMsgValue() (gas: 136384)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumInvalidVaa() (gas: 26781)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumSelfRedemptionWithValue() (gas: 39607)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumWrongEmitterAddress() (gas: 98141)
[32m[PASS][0m testCannotReceiveAndMintOnEthereumWrongEmitterChainId() (gas: 97392)
[32m[PASS][0m testCannotReceiveAndMintOnPolygon(uint16) (runs: 256, μ : 40400, \sim : 40400)
[32m[PASS][0m testCannotUpdateAmountsOnMintY00tsV2Deprecated() (gas: 17901)
[32m[PASS][0m testCannotUpdateAmountsOnMintY00tsV3OwnerOnly() (gas: 19506)
[32m[PASS][0m testCannotUpgradeY00tsV2OwnerOnly() (gas: 3238209)
[32m[PASS][0m testCannotUpgradeY00tsV3OwnerOnly() (gas: 3194497)
[32m[PASS][0m testDefaultRoyaltyY00tsV2() (gas: 24608)
[32m[PASS][0m testDefaultRoyaltyY00tsV3() (gas: 23942)
[32m[PASS][0m testForwardMessage(uint16) (runs: 256, μ : 349125, \sim : 349125)
[32m[PASS][0m testLockedY00tsV2() (gas: 109227)
[32m[PASS][0m testLockedY00tsV3() (gas: 105955)
[32m[PASS][0m testParseBatchPayload(uint256,uint256) (runs: 256, μ : 33678, \sim : 32878)
[32m[PASS][0m testReceiveAndMintBatch(uint256,uint256) (runs: 256, μ : 326175, \sim : 313337)
[32m[PASS][0m testReceiveAndMintOnEthereum(uint16) (runs: 256, μ : 233189, \sim : 233189)
[32m[PASS][0m testReceiveAndMintOnEthereumSelfRedemption(uint16) (runs: 256, μ : 154728, \sim : 154728)
[32m[PASS][0m testSupportsInterfaceY00tsV2() (gas: 15209)
[32m[PASS][0m testSupportsInterfaceY00tsV3() (gas: 14684)
[32m[PASS][0m testTokenRoyaltyY00tsV2() (gas: 35056)
[32m[PASS][0m testTokenRoyaltyY00tsV3() (gas: 34492)
[32m[PASS][0m testTokenURIOnEthereum(uint16) (runs: 256, μ : 237421, \sim : 237628)
[32m[PASS][0m testUpdateAmountsOnMintY00tsV3(uint256,uint256) (runs: 256, μ : 25727, \sim : 26102)
[32m[PASS][0m testUpgradeY00tsV2() (gas: 3242510)
[32m[PASS][0m testUpgradeY00tsV3() (gas: 3199454)
Test result: [32mok[0m. [32m56[0m passed; [31m0[0m failed; [33m0[0m skipped; finished in 1.68s

Ran 2 test suites: [32m57[0m tests passed, [31m0[0m failed, [33m0[0m skipped (57 total tests)
bash shell-scripts/run_integration_tests.sh
foundry.toml -> cache/foundry.toml
foundry-test.toml -> foundry.toml
deploying y00tsV3
overriding foundry.toml
cache/foundry.toml -> foundry.toml
Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))
Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))
Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))
Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

Environment Test

Global

✓ Environment Variables

Ethereum Goerli Testnet Fork

Environment

✓ Variables

RPC

✓ Wormhole (1033ms)

Avalanche Fuji Testnet Fork

Environment

✓ Variables

RPC

✓ Wormhole (60ms)

Polygon y00tsV2 Upgrade

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Deploy Implementation (452ms)

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Upgrade

Ethereum Migration

Test Forward From Solana to Ethereum

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Cannot Invoke `UpdateAmountsOnMint` On Polygon (Deprecated)

✓ Cannot Invoke `getAmountsOnMint` On Polygon (Deprecated)

✓ Cannot Invoke `ReceiveAndMint` On Polygon (Deprecated)

✓ Invoke `forwardMessage` On Polygon

✓ Invoke `receiveAndMint` on Ethereum With Forwarded Message (1383ms)

Test Migration From Polygon to Ethereum

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Invoke `burnAndSend` on Polygon

✓ Cannot Invoke `receiveAndMintBatch` (Batch Size == 1)

✓ Invoke `receiveAndMint` on Ethereum (38ms)

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Invoke `burnAndSend` With Batch on Polygon

✓ Cannot Invoke `receiveAndMint` on Ethereum (Batch Size == 4)

✓ Invoke `receiveAndMintBatch` on Ethereum (47ms)

Duplicate definition of Locked (Locked(uint256), Locked(address,address,uint256,uint256))

Duplicate definition of Unlocked (Unlocked(uint256), Unlocked(address,address,uint256))

✓ Invoke `burnAndSend` With Batch on Polygon (Max Batch Size) (104ms)

✓ Invoke `receiveAndMintBatch` on Ethereum (Max Batch Size) (143ms)

20 passing (3s)

7. Executive Summary

Two independent softstack experts performed an unbiased and isolated audit of the smart contract codebase provided by the Dust Labs Team. The main objective of the audit was to verify the security and functionality claims of the smart contract. The audit process involved a thorough manual code review and automated security testing.

The audit report provides detailed descriptions of each identified issue, including severity levels, CWE classifications, and recommendations for mitigation. It also includes code snippets, where applicable, to demonstrate the issues and suggest possible fixes. Notably, no issues or vulnerabilities were found during this comprehensive assessment.



8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, security, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over \$100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: <https://chainsulting.de>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.

