



DustLabs

Season 3 & Points Parlor

SMART CONTRACT AUDIT

09.08.2023

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 Source Unites in Scope	13
6. Scope of Work	14
6.1 Findings Overview	15
6.2 Manual and Automated Vulnerability Test.....	16
6.2.5 Unprotected Initializer	20
6.2.6 Missing Zero Address Checks	21
6.2.7 Withdraw Function Can Run Out Of Gas	21
6.2.8 Missing NatSpec Documentation.....	23
6.2.9 Floating Pragma	24
6.2.10 Documentation Mismatch	24



6.3 SWC Attacks	25
6.4 Verify Claims	28
6.4 Unit Tests	30
7. Executive Summary.....	32
8. About the Auditor	33

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 (07.08.2023)	Layout
0.4 (07.08.2023)	Automated Security Testing Manual Security Testing
0.5 (07.08.2023)	Verify Claims and Test Deployment
0.6 (08.08.2023)	Testing SWC Checks
0.9 (08.08.2023)	Summary and Recommendation
1.0 (09.08.2023)	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://degods.com>
<https://www.y00ts.com>

Twitter:

<https://twitter.com/DeGodsNFT>
<https://twitter.com/y00tsNFT>

Discord:

<https://discord.gg/dedao>
<https://discord.gg/y00ts>

Instagram:

<https://www.instagram.com/thedegods>
<https://instagram.com/they00ts>

The logo for y00ts, featuring the word "y00ts" in a stylized, handwritten black font.

DeGods

2.1 Project Overview

DeGods, originally launched on Solana, is an NFT collection celebrated for its distinctive generative art. Holders of these NFTs have the special capability to produce the DUST token, enhancing the collection's utility. The visionary behind DeGods is Rohun Vora, who, under the project, introduced the innovative "PHBT" mechanism, a strategy devised to uphold and potentially enhance the NFTs' market value. Recognizing the need for growth and expansion, DeGods took a pivotal step in December 2022 by migrating to Ethereum, a move that marked its evolution.

Not limiting themselves to Ethereum, the DeGods team, in February 2023, embraced the Ordinals protocol, a groundbreaking initiative that enabled digital asset representation on the Bitcoin blockchain. This strategic move into the Bitcoin ecosystem underscores DeGods' commitment to innovation and adaptability in the dynamic world of NFTs. The project's multi-chain approach, combined with its unique features, positions DeGods as a trailblazer in the NFT domain, setting new standards for digital collectibles.

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

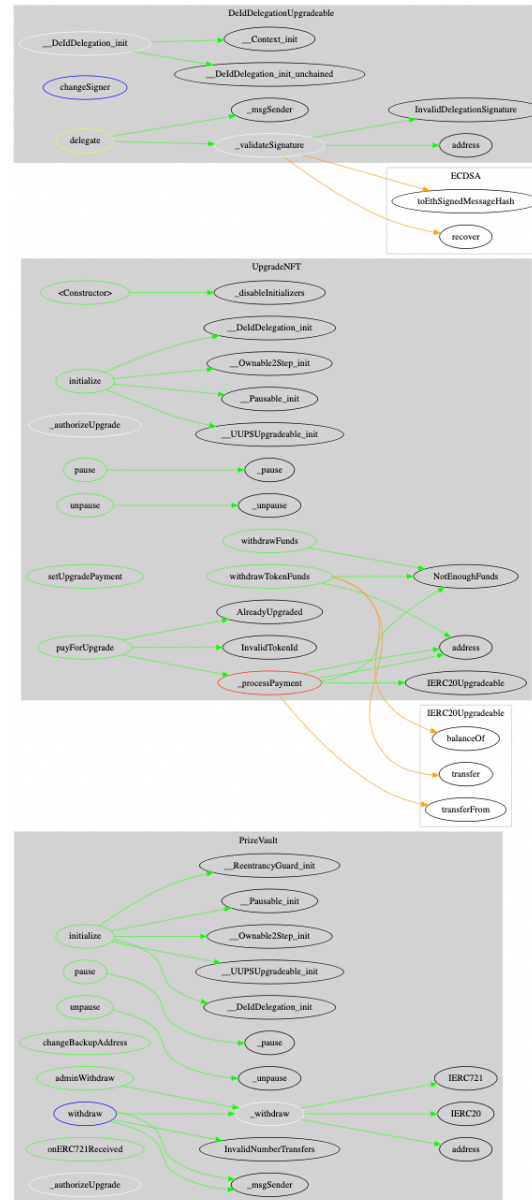
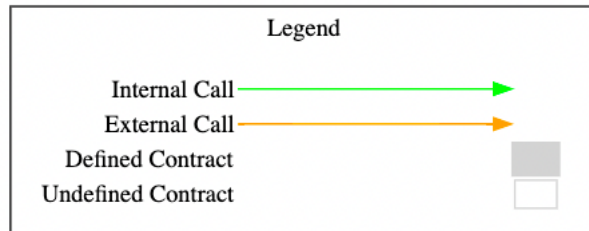
The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

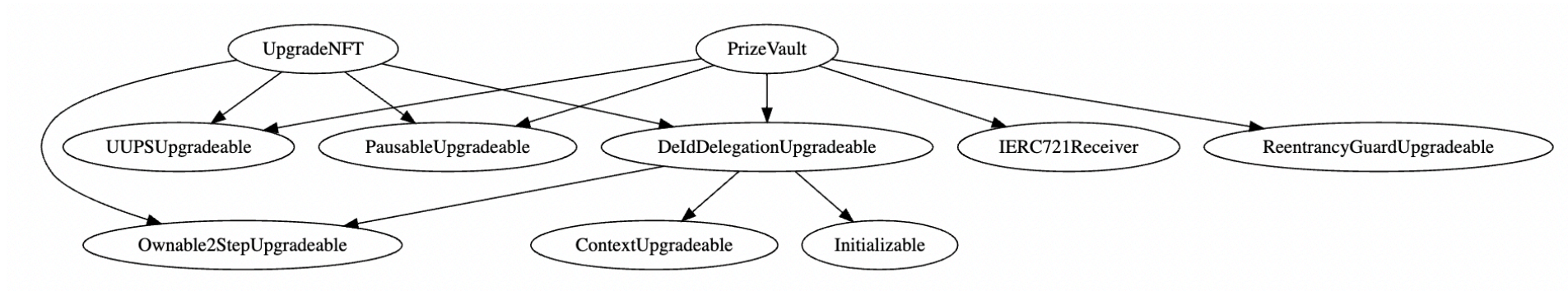
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability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

File	Fingerprint (MD5)
./PrizeVault.sol	0a362a736d9e917183e45ab47877f2f7
./lib/DeldDelegationUpgradeable.sol	44ff448d26791869a1939e83e6920262
./UpgradeNFT.sol	5c7354b41c3224c707b4e75b8c680fcf

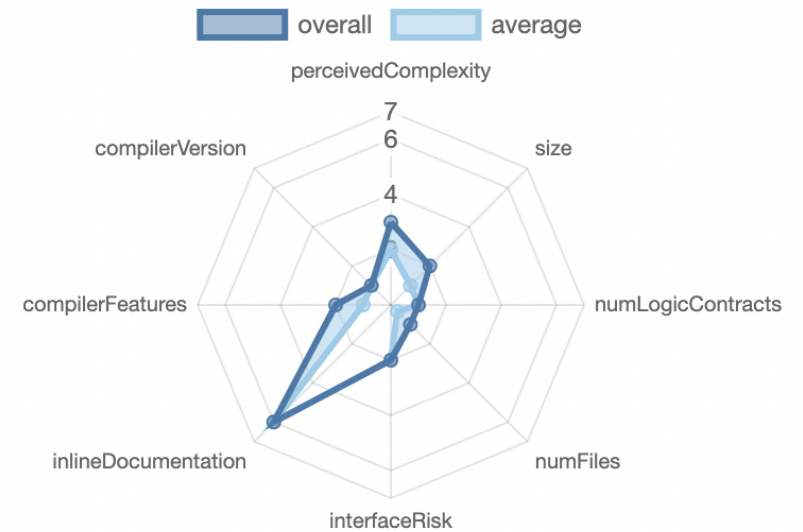
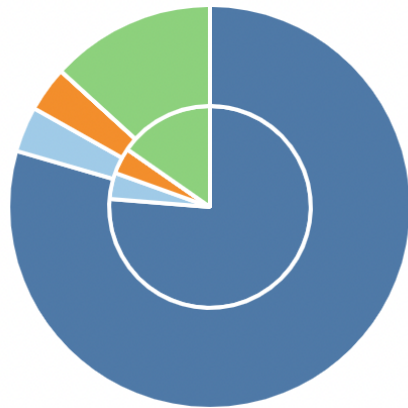
5.2 CallGraph



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
<div>0.8.19</div> <div>^0.8.0</div>			<div>yes</div>	<div></div>	<div></div>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECTrecover	 New/Create/Create2
<div>yes</div>	<div></div>	<div></div>	<div>yes</div>	<div></div>	

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				
15	1				
External	Internal	Private	Pure	View	
2	26	1	1	1	

StateVariables

Total	 Public
12	8

5.6 Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/PrizeVault.sol	1	_____	159	133	110	5	66	
	contracts/UpgradeNFT.sol	1	_____	133	116	91	2	75	
	contracts/lib/DeldDelegationUpgradable.sol	1	_____	79	70	53	7	29	
	Totals	3	_____	371	319	254	14	170	

- **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 Season 3 and Point Parlor contracts.

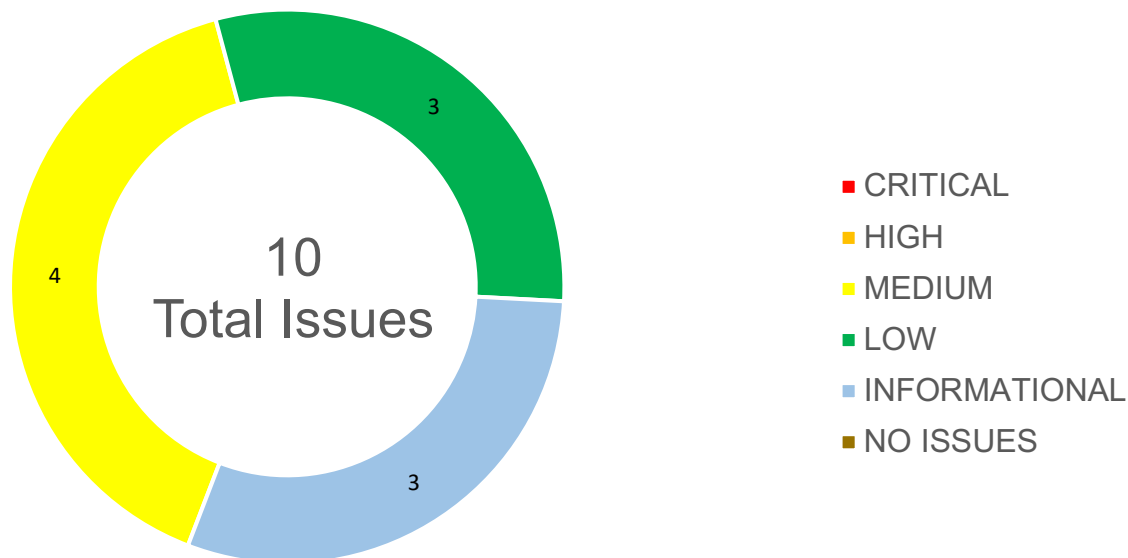
The team put forward the following assumptions regarding the security, usage of the contracts:

1. The security audit should meticulously examine the `UpgradeNFT` smart contract to ensure there are no vulnerabilities, potential exploits, or security flaws.
2. The integrity of the token upgrade mechanism must be validated, ensuring that tokens are accurately marked as upgraded and that the process is irreversible.
3. The delegation mechanism, as implemented by the `DeldDelegationUpgradeable` contract, should be scrutinized to confirm that only authorized transactions, signed by the designated signer, are processed.
4. The contract's fund withdrawal mechanisms, both for Ether and ERC20 tokens, should be audited to confirm that only the owner can access these funds and that the withdrawal processes are secure.
5. The effectiveness of the pausing mechanism, provided by the `PausableUpgradeable` feature, should be verified to ensure that the contract's main functionalities can be halted and resumed by the owner in case of emergencies or detected issues.

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



No	Title	Severity	Status
6.2.1	Transfer Ignores Return Value	MEDIUM	FIXED
6.2.2	Owner Can Withdraw Tokens (ERC20 and ERC721)	MEDIUM	ACKNOWLEDGED
6.2.3	Owner Can Pause Users Withdrawals	MEDIUM	ACKNOWLEDGED
6.2.4	Unchecked Return Value For Transfers	MEDIUM	FIXED
6.2.5	Unprotected Initializer	LOW	FIXED
6.2.6	Missing Zero Address Checks	LOW	ACKNOWLEDGED
6.2.7	Withdraw Function Can Run Out Of Gas	LOW	FIXED
6.2.8	Missing NatSpec Documentation	INFORMATIONAL	ACKNOWLEDGED
6.2.9	Floating Pragma	INFORMATIONAL	FIXED
6.2.10	Documentation Mismatch	INFORMATIONAL	FIXED

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 **4 Medium issues** in the code of the smart contract.

6.2.1 Transfer Ignores Return Value

Severity: MEDIUM

Status: FIXED

Code: CWE-252

File(s) affected: PrizeVault.sol

Attack / Description	The <i>_withdraw</i> function transfers arbitrary ERC20 and ERC721 token out of the contract. Both, the <i>transfer</i> function of ERC20 as well as the <i>transferFrom</i> function of ERC721 are returning a boolean, indicating if the transfer has been successful. If the token implementation is not reverting the function call on a transfer failure, the withdrawal will be successful without actually transferring the tokens.
Code	Line 87 - 98 (PrizeVault.sol) <pre>function _withdraw(address tokenAddress, PrizeType prizeType,</pre>

	<pre> address destination, uint256 tokenIdOrAmount) internal { if (prizeType == PrizeType.ERC20) { IERC20(tokenAddress).transfer(destination, tokenIdOrAmount); } else { IERC721(tokenAddress).transferFrom(address(this), destination, tokenIdOrAmount); } } </pre>
Result/Recommendation	It is recommended to check the return value of the <i>transfer</i> and <i>transferFrom</i> functions and revert the transaction if the return value is false.

6.2.2 Owner Can Withdraw Tokens (ERC20 and ERC721)

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: NA

File(s) affected: PrizeVault.sol

Attack / Description	By setting the <i>backupAddress</i> and calling <i>adminWithdraw</i> the owner of the contract can withdraw all ERC20 and ERC721 token hold by the contract at any time to an arbitrary address.
Code	<p>Line 100 - 107 (PrizeVault.sol)</p> <pre> function adminWithdraw(address tokenAddress, PrizeType prizeType, uint256 tokenIdOrAmount) public nonReentrant onlyOwner { _withdraw(tokenAddress, prizeType, backupAddress, tokenIdOrAmount); emit AdminWithdrawn(tokenAddress, prizeType, backupAddress, tokenIdOrAmount); } </pre>

	}
Result/Recommendation	It is recommended to remove the overpowered owner function <i>adminWithdraw</i> to reduce the risk of users not getting their prices on withdrawals. To recover uncollected prices, there could be a time limit to collect all prices. If the specified end timestamp is reached, the owner could be able to withdraw uncollected funds.

6.2.3 Owner Can Pause Users Withdrawals

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: NA

File(s) affected: PrizeVault.sol

Attack / Description	The contract implements the pausable functionality of OpenZeppelin, which allows the owner of the contract to pause all user withdrawals at any time by calling <i>pause</i> function due to the <i>whenNotPaused</i> modifier of <i>withdraw</i> function.
Code	<p>Line 109 - 117 (PrizeVault.sol)</p> <pre> function withdraw(TransferData calldata data, bytes calldata signature) external nonReentrant whenNotPaused delegate(_FUNC_NAME_WITHDRAW, abi.encode(data), signature) { </pre>
Result/Recommendation	It is recommended to remove the pausable functionality to reduce the degree of centralization and allow users at any time to withdraw their prices.

6.2.4 Unchecked Return Value For Transfers

Severity: MEDIUM

Status: FIXED

Code: CWE-252

File(s) affected: UpgradeNFT.sol

Attack / Description	The <i>_processPayment</i> function transfers an owner-defined ERC20 token from the caller to the contract. The <i>transfer</i> function is returning a boolean, indicating if the transfer has been successful. If the token implementation is not reverting the function call on a transfer failure, the upgrade will be successful without actually transferring the tokens. The same applies to the <i>withdrawTokenFunds</i> function.
Code	Line 91 - 102 (UpgradeNFT.sol) <pre>function _processPayment(uint256 tokenIdsLength, address paymentToken, uint256 price) private { uint256 totalPayment = price * tokenIdsLength; if (paymentToken == address(0)) { if (msg.value < totalPayment) { revert NotEnoughFunds(); } } else { IERC20Upgradeable token = IERC20Upgradeable(paymentToken); token.transferFrom(msg.sender, address(this), totalPayment); } }</pre>
Result/Recommendation	It is recommended to check the return value of the <i>transfer</i> function and revert the transaction if the return value is false.

LOW ISSUES

During the audit, softstack's experts found **3 Low issues** in the code of the smart contract

6.2.5 Unprotected Initializer

Severity: LOW

Status: FIXED

Code: NA

File(s) affected: PrizeVault.sol

Attack / Description	The logical contract of the PrizeVault has no protection against initialization by an attacker. It does not affect the functionality of the Proxy contract if it is called as it should be, but could lead to unintended behaviour if anybody is directly interacting with the logical contract.
Code	Line 61 - 72 (PrizeVault.sol) <pre>function initialize(uint256 chainId_, address signer_, address backupAddress_) public initializer { __UUPSUpgradeable_init(); __Ownable2Step_init(); __Pausable_init(); __ReentrancyGuard_init(); __DeIdDelegation_init(chainId_, signer_); backupAddress = backupAddress_; }</pre>
Result/Recommendation	It is recommended to call the <code>__disableInitializers</code> function of OpenZeppelins Initializer contract in the logical contracts constructor to avoid any unintended behaviour.

6.2.6 Missing Zero Address Checks

Severity: LOW

Status: ACKNOWLEDGED

Code: NA

File(s) affected: PrizeVault.sol

Attack / Description	The owner function <code>changeBackupAddress</code> does not check for the zero address on changing the address. If the address is set to the zero address and the owner calls <code>adminWithdraw</code> , funds are lost forever.
Code	Line 82 - 85 (PrizeVault.sol) <pre>function changeBackupAddress(address newBackupAddress) public onlyOwner { backupAddress = newBackupAddress; emit BackupAddressChanged(newBackupAddress); }</pre>
Result/Recommendation	It is recommended to check the <code>newBackupAddress</code> to be non zero in a <code>require</code> statement to prevent the unintended permanent loss of funds.

6.2.7 Withdraw Function Can Run Out Of Gas

Severity: LOW

Status: FIXED

Code: NA

File(s) affected: PrizeVault.sol

Attack / Description	The <i>withdraw</i> function is looping over <i>data.details</i> array to process withdrawals. If the size of this
-----------------------------	--

	array is large, it can potentially lead to the transaction running out of gas.
Code	<p>Line 109 - 141 (PrizeVault.sol)</p> <pre> function withdraw(TransferData calldata data, bytes calldata signature) external nonReentrant whenNotPaused delegate(_FUNC_NAME_WITHDRAW, abi.encode(data), signature) { if (data.details.length == 0) { revert InvalidNumberTransfers(); } for (uint256 i = 0; i < data.details.length;) { TransferDetail memory detail = data.details[i]; _withdraw(detail.contractAddress, detail.prizeType, data.recipient, detail.tokenIdOrAmount); emit PrizeWithdrawn(detail.contractAddress, detail.prizeType, _msgSender(), data.recipient, detail.tokenIdOrAmount, nonce[_msgSender()]); } } </pre>

	<pre> unchecked { i++; } } } } </pre>
Result/Recommendation	To prevent the contract from running out of gas, consider introducing a maximum limit to the length of <i>data.details</i> .

INFORMATIONAL ISSUES

During the audit, softstack's experts found **3 Informational issues** in the code of the smart contract.

6.2.8 Missing NatSpec Documentation

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: PrizeVault.sol, DeldDelegationUpgradeable.sol, UpgradedNFT.sol

Attack / Description	Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).
Code	NA
Result/Recommendation	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

--	--

6.2.9 Floating Pragma

Severity: INFORMATIONAL

Status: FIXED

Code: SWC-103

File(s) affected: UpgradeNFT.sol

Attack / Description	The current pragma Solidity directive is ^0.8.0; It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
Code	Line 2 (UpgradeNFT.sol) <code>pragma solidity ^0.8.0;</code>
Result/Recommendation	It is recommended to follow the example (0.8.19), as future compiler versions may handle certain language constructions in a way the developer did not foresee. Not effecting the overall contract functionality.

6.2.10 Documentation Mismatch

Severity: INFORMATIONAL

Status: FIXED

Code: NA

File(s) affected: UpgradeNFT.sol

Attack / Description	The documentation mentions a function <i>payForGenderSwitch</i> which is not implemented in the code. This could lead to confusion for users, developers or auditors. Additionally, the <i>female</i> variable in the <i>TokenInfo</i> struct is useless in the current implementation.
-----------------------------	---



Code	<p>In README-Documentation: The contract has the following functions:</p> <ul style="list-style-type: none"> - <code>`payForUpgrade`</code>: Upgrades the NFTs and sets their gender to Male by default. An event <code>`NFTUpgraded`</code> is emitted. - <code>`payForGenderSwitch`</code>: Switches the gender of the NFTs. An event <code>`SwitchedGender`</code> is emitted.)
Result/Recommendation	Update the documentation and remove the <i>TokenInfo.female</i> variable to reflect the actual functionalities provided by the smart contract or implement the <i>payForGenderSwitch</i> function if it is required.

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	✓

ID	Title	Relationships	Test Result
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	✓
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	✓

ID	Title	Relationships	Test Result
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')	✓
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	✓

ID	Title	Relationships	Test Result
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	✓
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 The security audit should meticulously examine the `UpgradeNFT` smart contract to ensure there are no vulnerabilities, potential exploits, or security flaws.

Status: tested and verified ✓

6.4.2 The integrity of the token upgrade mechanism must be validated, ensuring that tokens are accurately marked as upgraded and that the process is irreversible.

Status: tested and verified ✓

6.4.3 The delegation mechanism, as implemented by the `DeldDelegationUpgradeable` contract, should be scrutinized to confirm that only authorized transactions, signed by the designated signer, are processed.

Status: tested and verified ✓

6.4.4 The contract's fund withdrawal mechanisms, both for Ether and ERC20 tokens, should be audited to confirm that only the owner can access these funds and that the withdrawal processes are secure.

Status: tested and verified ✓

6.4.5 The effectiveness of the pausing mechanism, provided by the `PausableUpgradeable` feature, should be verified to ensure that the contract's main functionalities can be halted and resumed by the owner in case of emergencies or detected issues.

Status: tested and verified ✓

6.4 Unit Tests

PrizeVault test

Owner

- ✓ should have the correct initial owner
- ✓ should have the correct initial signer
- ✓ should have the correct initial backup address
- ✓ should revert when non-owner assign new owner
- ✓ should revert when non-owner changes the signer
- ✓ should revert when non-owner changes the backup address
- ✓ should revert when non-owner tries to pause
- ✓ should revert when non-owner tries to unpause
- ✓ should be able to transfer ownership
- ✓ owner should be able to change the signer
- ✓ owner should be able to change the backup address
- ✓ owner should be able to pause & unpause (45ms)

Withdraw

- ✓ should revert with empty transfer details list
- ✓ should revert with withdraw to invalid recipient
- ✓ should revert when nonce mismatch
- ✓ should revert when signature is invalid
- ✓ should revert when making invalid transfer
- ✓ should revert when non-owner withdraw as admin
- ✓ should revert when withdraw during pause
- ✓ should be able to withdraw (64ms)
- ✓ should be able to withdraw with increased nonce (43ms)
- ✓ should be able to withdraw as admin

22 passing (2s)

File ▲		Statements ▾		Branches ▾		Functions ▾		Lines ▾	
contracts/	<div></div>	100%	19/19	79.17%	19/24	88.89%	8/9	100%	24/24
contracts/lib/	<div></div>	100%	10/10	75%	6/8	100%	5/5	100%	17/17

UpgradeNFT



Owner

- ✓ Should set owner to deployer
- ✓ Should allow owner to pause
- ✓ Should allow owner to unpause
- ✓ Should not allow nonOwner to pause
- ✓ Should not allow nonOwner to unpause
- ✓ Should allow owner to set upgrade fee
- ✓ Should not allow non-owner to set upgrade fee

Token upgrade & gender switch with ERC20

- ✓ Should upgrade token
- ✓ Should upgrade multiple tokens
- ✓ Should not allow token to be upgraded twice
- ✓ Should not allow the upgrade if one of the tokens has already been upgraded
- ✓ Should not allow token to be upgraded with insufficient funds
- ✓ Should not allow a token upgrade if one of the tokenIds is outside of the range

13 passing (2s)

File ▲		Statements ⇅		Branches ⇅		Functions ⇅		Lines ⇅	
contracts/		79.17%	19/24	46.67%	14/30	70%	7/10	77.78%	28/36
contracts/lib/		90%	9/10	37.5%	3/8	80%	4/5	82.35%	14/17

7. Executive Summary

Three 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.

Overall, the audit identified a total of 10 issues, classified as follows:

- No critical issues were found.
- No high severity issues were found.
- Four medium severity issues were found.
- Three low severity issues were discovered
- Three informational issues were identified

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. We advise the Dust Labs team to implement the recommendations to further enhance the code's security and readability.

Update (09.08.2023): All necessary fixes have been applied.



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.

