QuillAudits

Audit Report
November, 2020

Contents

Introduction

This Audit Report mainly focuses on the overall security of Digitalax Smart Contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Quillhash team has performed rigorous testing of the project starting with analysing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.

In Automated Testing, We tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analysing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Audit Details

Project Name: Digitalax
Website/Etherscan Code: https://www.digitalax.xyz/homepage
Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security
Identifying security related issues within each contract and the system of contract.

Sound Architecture
Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality
A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	1	2	9
Closed	1	3	3	1

Manual Audit

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM, Rinkeby and Kovan networks to test the contract functionality.

Contracts were tested on Kovan and Remix JVM. Heavy testing is done on the parent child relationship and Auction contracts flow:

Helper to calculate bytes form of token IDs:

0x5F00928a589DA94bC9Df8B073f8d4FA9616cc1eB

DIGITALAXGARMENTFACTORY: 0xc21D8A2ed5dC5F0004e2De35931310C6385df543

DIGITALAXAUCTION: 0xc21D8A2ed5dC5F0004e2De35931310C6385df543

DIGITALAXGARMENTNFT: 0xb6162505F957B43b5b23a071d5492018bec53Edd

DIGITALAXMATERIALS: 0xEfC12bcF85bd0D770a157572FD3D768472EE3eE7

DIGITALAXACCESSCONTROLS: 0x1c68816b03B97f8aD4057Ba5a41e32e8C9ad0e75

Other instances of Kovan testing:

DIGITALAXGARMENTNFT : 0xde951D5445d66310aF9ad6DA03cA47bB706c0c9B

DIGITALAXAUCTION:0x0369296a594f77D3cF98c87A360d38276aDa7c9C

ERC1155: 0x1d4f2f4258e7fce8a90b5873a99331d4064676a5

DIGITALAXACCESSCONTROLS: 0x07e5838f3fcc0c0363b55e27be76b6ae0b22c9eb

DIGITALAXGARMENTNFT : 0x42e313b502b79bd2a57850fe5425e621440fc7a1

DIGITALAXAUCTION: 0x2d043db2d2cb615d1bed82ec7163798875eacb0d

ERC1155: 0x611961Ca9B91adA5911beAEfF2F7ce55eA037Cc6

DIGITALAXMATERIALS: 0x0c0e5cafb3f795f69b94c70e013212bf3d0b51ed

DIGITALAXGARMENTFACTORY: 0xfa6b5c3fb5e3c6266b278f33da324b8b1fb81677

Unit Test Cases: DONE

DigitalaxMaterials.sol

1. Test if contract fails to deploy with incorrect list of arguments
2. Test if contract can be deployed when correct list of arguments - name, symbol,accesscontrol("DigitalaxMaterials", "DXM", accessControlsAddress) are passed.
3. Test if access controls can be updated as admin
   - when sender is admin and smart contract
   - when sender is not current access control address
   - when sender is not zero address.

4. Test if update access control fails when sender is zero address
5. Test if update access control fails when sender is not admin
6. Test if update access control fails when sender is current access control address
7. Test if update access control fails when sender is not a smart contract
8. Test if a single child ERC1155 token can be created when sender is smart contract and URI is not blank
9. Test if a single child ERC1155 token cannot be created when sender is not a smart contract
10. Test if a single child ERC1155 token cannot be created when URI is a blank string
11. Test if token id gets generated after successfully creating a single child ERC1155 token
12. Test if a batch of child ERC1155 tokens can be created when sender is smart contract, array of URIs is not blank and length of each element of uri array is not zero
13. Test if a batch of child ERC1155 tokens cannot be created when sender is not a smart contract
14. Test if a batch of child ERC1155 tokens cannot be created when any URI in the array is a blank string
15. Test if a batch of child ERC1155 tokens cannot be created when uri array is empty
16. Test if token ids gets generated after successfully creating a batch of child ERC1155 tokens
17. Test if mints a single child ERC1155 token when sender is a smart contract, strand provided exists and amount is not zero.
18. Test if minting a single child ERC1155 token fails when sender is not a smart contract
19. Test if minting a single child ERC1155 token fails when sender strand provided does not exists
20. Test if minting a single child ERC1155 token fails when amount is zero.
21. Test if mints a batch of children ERC1155 tokens when sender is a smart contract, array of children token ids is not empty, array of amounts is not empty.
22. Test if minting of batch of children ERC1155 tokens fails when there is a mismatch in number of elements in array of children token ids and array of amounts.
23. Test if minting of batch of children ERC1155 tokens fails when sender is not a smart contract
24. Test if minting of batch of children ERC1155 tokens fails when array of token ids is empty.
25. Test if minting of batch of children ERC1155 tokens fails when array of amounts is empty.
26. Test if minting of batch of children ERC1155 tokens fails when an element of array of token ids is blank.
27. Test if minting of batch of children ERC1155 tokens fails when an element of array of amounts is blank or any amount in array is zero.
28. Test if minting of batch of children ERC1155 tokens fails when any token id does not exist

DigitalaxGarmentFactory.sol

1. Test if contract fails to deploy with incorrect list of arguments
2. Test if contract can be deployed with correct list of arguments - name, symbol,accesscontrol(garmentAddress, materialsAddress, accessControlsAddress)
3. Test if a single child ERC1155 token can be created when sender has minter role
4. Test if a single child ERC1155 token cannot be created when sender does not have a minter role.
5. Test if uri gets generated after successfully creating a strand
6. Test if a multiple strands get created when sender has minter role
7. Test if uris gets generated after successfully creating a batch of strands
8. Test if a single ERC721 garment parent token get generated, parent token id is minted along with a batch of ERC1155 child tokens and when sender has minter role and an array of strand uris are passed.
9. mintParentWithChildren: Test if a single ERC721 garment parent token cannot be generated when sender does not has minter role
10. mintParentWithChildren:Test if a single ERC721 garment parent token cannot be generated when an array of strand uris are not passed properly.
11. Test if a single ERC721 garment parent token get generated without linked child tokens and when sender has minter role
12. Test if a single ERC721 garment parent token does not get generated without linked child tokens and when sender does not has minter role

Low level severity issues

1. Multiplication after division: The instance below performs a multiplication operation after division:

```
1518            uint256 platformFeeAboveReserve = (aboveReservePrice.div(1000)).mul(platformFee);
1519
```

   By doing all our multiplications first, we mitigate rounding related issues as much as possible. The computations can be much more complex and forming because Solidity operates only with integers. Thus, if the division is done before the multiplication, the rounding errors can increase dramatically.

   For Instance:

   In the following example, `amount` variable is divided by `DELIMITER` and then multiplied by `BONUS` . Thus, a rounding error appears (consider `amount = 9000`):

```
pragma solidity 0.4.25;
contract MyContract {
    uint constant BONUS = 500;
    uint constant DELIMITER = 10000;

    function calculateBonus(uint amount) returns (uint) {
        return amount/DELIMITER*BONUS;
    }
}
```

   Fixed

2. There are multiple linting violations within the contract, we recommend using solhint to fix them.

```
        auction.reservePrice,
        auction.startTime,
        auction.endTime,
        auction.resulted
            auction.reservePrice,
            auction.startTime,
            auction.endTime,
            auction.resulted
```

3. The pragama versions used within some contracts are not locked:

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.6.0;
```

```
pragma solidity ^0.7.0; // bad: compiles with 0.7.0 and above
pragma solidity 0.7.0; // good : compiles w 0.7.0 only
```

It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee.

Fixed

4. DigitalaxMaterials.sol: Use local variable instead of state variable like .length in a loop [#87-94, 141-147]

   For every iteration of for loop - state variables like .length of non-memory array will consume extra gas. To reduce the gas consumption it's advisable to use a local variable.

5. DigitalaxMaterials.sol: Be explicit about which `uint` the code is using[#87,141]

   `uint` is an alias for `uint256`, but using the full form is preferable. Be consistent and use one of the forms.

   Fixed

Medium severity issues

1. Duplicate Code: Function _asSingletonArray within DigitalaxGarmentNFT is a private function which is an exact replica of _asSingletonArray function found within ERC1155 contracts. It has not been used anywhere within the DigitalaxGarmentNFT contract.

```
    function _asSingletonArray(uint256 element) private pure returns (uint256[] memory) {
        uint256[] memory array = new uint256[](1);
        array[0] = element;

        return array;
    }
```

Fixed

2. State variable .length of non-memory array is used in the condition of for loop at multiple occurrences. In this case, every iteration of the loop consumes extra gas.

```
2919            for (uint256 i = 0; i < parentToChildMapping[_tokenId].length(); i++) {
2920                childTokenIds[i] = parentToChildMapping[_tokenId].at(i);
2921            }
```

If the state variable .length is used several times, holding its value in a local variable is more gas efficient. Though if .length of the calldata-array is placed into a local variable, the optimisation will be less significant.

For instance:

In the following example, `limiter` variable is accessed on every for-loop iteration:

```
pragma solidity 0.4.25;

contract NewContract {
    uint limiter = 100;

    function longLoop() {
        for(uint i = 0; i < limiter; i++) {
            /* ... */
        }
    }
}
```

3. The following list of functions should be made external for saving gas during function calls:

setApprovalForAll(address,bool):

ERC1155.setApprovalForAll(address,bool)

safeTransferFrom(address,address,uint256,uint256,bytes):

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes)

safeBatchTransferFrom(address,address,uint256[],uint256[],bytes):

ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes)

onERC1155Received(address,address,uint256,uint256,bytes):

DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)

onERC1155BatchReceived(address,address,uint256[],uint256[],bytes):

DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)

The functions below are from standard Open Zeppelin and therefore can be omitted:

renounceRole(bytes32,address):
AccessControl.renounceRole(bytes32,address)
setApprovalForAll(address,bool):
ERC721.setApprovalForAll(address,bool)
transferFrom(address,address,uint256):
ERC721.transferFrom(address,address,uint256)
safeTransferFrom(address,address,uint256):
ERC721.safeTransferFrom(address,address,uint256)

Note: There are four types of visibilities for functions and state variables. Functions can be specified as being external, public, internal or private, where the default is public. For state variables, external is not possible and the default is internal.

Fixed

4. DigitalaxGarmentFactory.sol: For DigitalaxGarmentFactory contract implement ReentrancyGuard logic as done in DigitalaxAuction.sol. It's recommended that the calls to external functions/events should happen after any changes to state variables in your contract so your contract is not vulnerable to a re-entrancy exploit. To avoid re-entrancy exploit make use of Openzeppelin ReentrancyGuard.sol.
Lines: #72-91, #97-112, #221-262.

Fixed

High severity issues

1. Business Logic Recommendation: A malicious bidder can bid and withdraw his funds just before a sale ends to make sure an auction is cancelled. If Digitalax wants to have their bidders with an option to withdraw their bids, we recommend placing a check to restrict withdrawals when an auction is close to its end.

Mitigation strategy has been discussed with the Digitalax team.

Informational

1. Minor Readability Recommendation: DigitalaxAccessControl: Modifier is a reserved keyword in Solidity. The comment though not wrong can be confusing, it would be good to rename the comment to “Setters” OR “Updaters”.

```
////////////////
// Modifiers //
////////////////
```

2. Gas Optimization Recommendation: Following events can be removed if not absolutely necessary:
    - AdminRoleGranted
    - AdminRoleRemoved
    - MinterRoleGranted
    - MinterRoleRemoved
    - SmartContractRoleGranted
    - SmartContractRoleRemoved

    While it is good to emit an event whenever a state is changed within the contract the following events are duplicate. As the corresponding state changes are already covered by following OpenZeppelin events:
    - RoleGranted
    - RoleRevoked

    Removing them can save contract deployment cost as well as execution cost.

3. Business Logic Recommendation: Because of the following check within the DigitalaxAuction contract "auctions[_garmentTokenId].endTime == 0" you cannot re-auction an NFT whose auction has ended. Consider a scenario when a bidder who bought an item wants to re-auction the bought garment at a later date. Because of this check, it won't be possible.

```
function _createAuction(
    uint256 _garmentTokenId,
    uint256 _reservePrice,
    uint256 _startTimestamp,
    uint256 _endTimestamp
) private {
    // Ensure a token cannot be re-listed if previously successfully sold
    require(auctions[_garmentTokenId].endTime == 0, "DigitalaxAuction.createAuction: Cannot relist");
```

4. Many *require* checks within the contract are repetitive except their log statements, they can be turned into modifiers and hence redundancy can be reduced.

5. IERC998ERC1155TopDown is not a standard implementation of the reference provided. Some functions have not been implemented and event TransferSingleChild though implemented in the interface has never been emitted within the contracts. It is recommended to tweak interface name as it is used for inheritance within DigitalaxGarmentNFT contract.

   Fixed

6. Block timestamps can be manipulated by miners, too much dependence for high value assets can be exploited to miners advantage to gain favour in the bidding process.

7. DOS attack vector within the auction contract: When the auction is created, the ownership of the NFT is not transferred to the auction contract. Therefore before the sale is resulted the NFT owner can transfer the asset to some other address OR remove approval of the auction contract to act as an operator. This will result in failure to call resultAuction function, hence failing to finalize the sale. It is recommended to transfer NFT asset ownership to the auction contract during the creation to avoid this scenario.

8. There should be a require statement within addSmartContract function to check if the msg.sender is a contract, we suggest using isContract function of OpenZeppelin's Address Library.

9. Replace multiple return values with a struct

```
1737      function getAuction(uint256 _garmentTokenId)
1738      external
1739      view
1740      returns (uint256 _reservePrice, uint256 _startTime, uint256 _endTime, bool _resulted) {
1741          Auction storage auction = auctions[_garmentTokenId];
1742          return (
1743          auction.reservePrice,
1744          auction.startTime,
1745          auction.endTime,
1746          auction.resulted
1747          );
1748      }
```

Consider using struct instead of multiple return values for internal or private functions. It can improve code readability.

Automated Testing

Remix IDE

Remix compiler did not throw any warning for the given set of contracts.

SmartCheck

Smartcheck is a tool for automated static analysis of Solidity source code for security vulnerabilities and best practices. SmartCheck translates Solidity source code into an XML-based intermediate representation and checks it against XPath patterns. Smartcheck shows significant improvements over existing alternatives in terms of false discovery rate (FDR) and false negative rate (FNR). It gave the following result for the Digitalax contracts:

Flattened version of DigitalaxGarmentNFT:
https://tool.smartdec.net/scan/0e468f426338422d8fbb1570d0320445
Flattened version of DigitalaxAuction:
https://tool.smartdec.net/scan/39bb8347ed754a20a7fad6552368e35c
Flattened version of DigitalaxAccessControls:
https://tool.smartdec.net/scan/ec1a0eb9a4a34bedab43d87a718d3378
Flattened version of ERC1155:
https://tool.smartdec.net/scan/7066899744074f5da82952dab86c305e
Flattened version of DigitalaxMaterials.sol and DigitalaxGarmentFactory.sol:
https://tool.smartdec.net/scan/245e9eafeaa249b1a1885cb5457c559a

SmartCheck did not detect any high severity issue. All the considerable issues raised by SmartCheck are already covered in the Manual Audit section of this report.

Slither

Slither is an open-source Solidity static analysis framework. This tool provides rich information about Ethereum smart contracts and has the critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.

```
~/work/projects/audits/digitalax-contracts/smart-contracts   main 3  slither .
'npx truffle compile --all' running (use --truffle-version truffle@x.x.x to use specific version)

Compiling your contracts...
===========================
> Compiling ./contracts/DigitalaxAccessControls.sol
> Compiling ./contracts/DigitalaxAuction.sol
> Compiling ./contracts/DigitalaxGenesisNFT.sol
> Compiling ./contracts/ERC1155/ERC1155.sol
> Compiling ./contracts/ERC1155/ERC1155Burnable.sol
> Compiling ./contracts/ERC721/ERC721WithSameTokenURIForAllTokens.sol
> Compiling ./contracts/ERC998/IERC998ERC1155TopDown.sol
> Compiling ./contracts/garment/DigitalaxGarmentFactory.sol
> Compiling ./contracts/garment/DigitalaxGarmentNFT.sol
> Compiling ./contracts/garment/DigitalaxMaterials.sol
> Compiling ./contracts/garment/IDigitalaxGarmentNFT.sol
> Compiling ./contracts/mock/AlwaysRevertingEthReceiver.sol
> Compiling ./contracts/mock/BiddingContractMock.sol
> Compiling ./contracts/mock/DigitalaxAuctionMock.sol
> Compiling ./contracts/mock/DigitalaxGenesisNFTMock.sol
> Compiling ./contracts/mock/ERC1155Mock.sol
> Compiling ./contracts/mock/ERC1155ReceiverMock.sol
> Compiling ./contracts/mock/ERC165Mock.sol
> Compiling ./contracts/mock/ERC721ReceiverMock.sol
> Compiling ./contracts/mock/MockERC20.sol
> Compiling ./contracts/mock/MockVault.sol
> Compiling @openzeppelin/contracts/GSN/Context.sol
> Compiling @openzeppelin/contracts/access/AccessControl.sol
> Compiling @openzeppelin/contracts/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC1155/ERC1155Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155MetadataURI.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/EnumerableMap.sol
> Compiling @openzeppelin/contracts/utils/EnumerableSet.sol
> Compiling @openzeppelin/contracts/utils/ReentrancyGuard.sol
> Compiling @openzeppelin/contracts/utils/Strings.sol
> Compilation warnings encountered:

/home/ayush/work/projects/audits/digitalax-contracts/smart-contracts/contracts/DigitalaxGenesisNFT.sol:
260:5: Warning: Function state mutability can be restricted to pure
    function _getMaxGenesisContributionTokens() internal virtual view returns (uint256) {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to /home/ayush/work/projects/audits/digitalax-contracts/smart-contracts/build/contracts
> Compiled successfully using:
   - solc: 0.6.12+commit.27d51765.Emscripten.clang

INFO:Detectors:
DigitalaxAuction.resultAuction(uint256) (DigitalaxAuction.sol#315-420) sends eth to arbitrary user
    Dangerous calls:
    - (platformTransferSuccess) = platformFeeRecipient.call{value: platformFeeAboveReserve}()
```

```
(DigitalaxAuction.sol#385-387)
	- (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value:
winningBid.sub(platformFeeAboveReserve)}() (DigitalaxAuction.sol#394-396)
	- (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value: winningBid}()
(DigitalaxAuction.sol#403-405)
DigitalaxAuction._refundHighestBidder(address,uint256) (DigitalaxAuction.sol#745-758) sends eth to arbitrary
user
	Dangerous calls:
	- (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}() (DigitalaxAuction.sol#750-752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-
arbitrary-destinations
INFO:Detectors:
Reentrancy in DigitalaxAuction.cancelAuction(uint256) (DigitalaxAuction.sol#427-463):
	External calls:
	- _refundHighestBidder(highestBid.bidder,highestBid.bid) (DigitalaxAuction.sol#453)
		- (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}()
(DigitalaxAuction.sol#750-752)
	State variables written after the call(s):
	- delete auctions[_garmentTokenId] (DigitalaxAuction.sol#460)
	- delete highestBids[_garmentTokenId] (DigitalaxAuction.sol#456)
Reentrancy in DigitalaxAuction.placeBid(uint256) (DigitalaxAuction.sol#221-262):
	External calls:
	- _refundHighestBidder(highestBid.bidder,highestBid.bid) (DigitalaxAuction.sol#253)
		- (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}()
(DigitalaxAuction.sol#750-752)
	State variables written after the call(s):
	- highestBid.bidder = _msgSender() (DigitalaxAuction.sol#257)
	- highestBid.bid = bidAmount (DigitalaxAuction.sol#258)
	- highestBid.lastBidTime = _getNow() (DigitalaxAuction.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
DigitalaxAuction.resultAuction(uint256) (DigitalaxAuction.sol#315-420) performs a multiplication on the result
of a division:
	-platformFeeAboveReserve = (aboveReservePrice.div(1000)).mul(platformFee)
(DigitalaxAuction.sol#380-382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Contract locking ether found in :
	Contract AlwaysRevertingEthReceiver (mock/AlwaysRevertingEthReceiver.sol#5-9) has payable functions:
	 - AlwaysRevertingEthReceiver.receive() (mock/AlwaysRevertingEthReceiver.sol#6-8)
	But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
Reentrancy in DigitalaxGarmentNFT.burn(uint256) (garment/DigitalaxGarmentNFT.sol#105-124):
	External calls:
	- _extractAndTransferChildrenFromParent(_tokenId,_msgSender()) (garment/DigitalaxGarmentNFT.sol#115)
		- childContract.safeBatchTransferFrom(address(this),_to,_childTokenIds,_amounts,abi.encodePacked())
(garment/DigitalaxGarmentNFT.sol#346)
	State variables written after the call(s):
	- _burn(_tokenId) (garment/DigitalaxGarmentNFT.sol#119)
		- _tokenApprovals[tokenId] = to (@openzeppelin/contracts/token/ERC721/ERC721.sol#453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (ERC1155/
ERC1155.sol#365) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response
(ERC1155/ERC1155.sol#384) is a local variable never initialized
DigitalaxAuction.resultAuction(uint256).designerTransferSuccess_scope_0 (DigitalaxAuction.sol#403) is a local
variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason
(ERC1155/ERC1155.sol#388) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (ERC1155/
ERC1155.sol#361) is a local variable never initialized
```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ERC721WithSameTokenURIForAllTokens._mint(address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#311-322) ignores return value by _holderTokens[to].add(tokenId)
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#317)
ERC721WithSameTokenURIForAllTokens._mint(address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#311-322) ignores return value by _tokenOwners.set(tokenId,to)
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#319)
ERC721WithSameTokenURIForAllTokens._burn(uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#334-347) ignores return value by
_holderTokens[owner].remove(tokenId) (ERC721/ERC721WithSameTokenURIForAllTokens.sol#342)
ERC721WithSameTokenURIForAllTokens._burn(uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#334-347) ignores return value by _tokenOwners.remove(tokenId)
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#344)
ERC721WithSameTokenURIForAllTokens._transfer(address,address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#360-375) ignores return value by
_holderTokens[from].remove(tokenId) (ERC721/ERC721WithSameTokenURIForAllTokens.sol#369)
ERC721WithSameTokenURIForAllTokens._transfer(address,address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#360-375) ignores return value by _holderTokens[to].add(tokenId)
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#370)
ERC721WithSameTokenURIForAllTokens._transfer(address,address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#360-375) ignores return value by _tokenOwners.set(tokenId,to)
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#372)
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ERC1155/
ERC1155.sol#350-371) ignores return value by
IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (ERC1155/ERC1155.sol#361-369)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ERC1155/
ERC1155.sol#373-394) ignores return value by
IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (ERC1155/
ERC1155.sol#384-392)
MockVault.mintAssetBackedSyntheticMaterial(uint256) (mock/MockVault.sol#22-33) ignores return value by
supportedERC20Asset.transferFrom(msg.sender,address(this),_depositAmount) (mock/MockVault.sol#24)
MockVault.mintAssetBackedSyntheticMaterialToGarment(address,uint256,uint256) (mock/
MockVault.sol#35-48) ignores return value by
supportedERC20Asset.transferFrom(msg.sender,address(this),_depositAmount) (mock/MockVault.sol#39)
MockVault.claimUnderlyingAssetFromMaterial(uint256,uint256) (mock/MockVault.sol#50-54) ignores return
value by supportedERC20Asset.transfer(msg.sender,_claimAmount) (mock/MockVault.sol#53)
ERC721._mint(address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#333-344) ignores return
value by _holderTokens[to].add(tokenId) (@openzeppelin/contracts/token/ERC721/ERC721.sol#339)
ERC721._mint(address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#333-344) ignores return
value by _tokenOwners.set(tokenId,to) (@openzeppelin/contracts/token/ERC721/ERC721.sol#341)
ERC721._burn(uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#356-374) ignores return value by
_holderTokens[owner].remove(tokenId) (@openzeppelin/contracts/token/ERC721/ERC721.sol#369)
ERC721._burn(uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#356-374) ignores return value by
_tokenOwners.remove(tokenId) (@openzeppelin/contracts/token/ERC721/ERC721.sol#371)
ERC721._transfer(address,address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#387-402)
ignores return value by _holderTokens[from].remove(tokenId)
(@openzeppelin/contracts/token/ERC721/ERC721.sol#396)
ERC721._transfer(address,address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#387-402)
ignores return value by _holderTokens[to].add(tokenId) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#397)
ERC721._transfer(address,address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#387-402)
ignores return value by _tokenOwners.set(tokenId,to) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#399)
DigitalaxGarmentNFT._receiveChild(uint256,address,uint256,uint256) (garment/
DigitalaxGarmentNFT.sol#351-356) ignores return value by
parentToChildMapping[_tokenId].add(_childTokenId) (garment/DigitalaxGarmentNFT.sol#353)
DigitalaxGarmentNFT._removeChild(uint256,address,uint256,uint256) (garment/
DigitalaxGarmentNFT.sol#358-365) ignores return value by
childToParentMapping[_childTokenId].remove(_tokenId) (garment/DigitalaxGarmentNFT.sol#362)
DigitalaxGarmentNFT._removeChild(uint256,address,uint256,uint256) (garment/
DigitalaxGarmentNFT.sol#358-365) ignores return value by

parentToChildMapping[_tokenId].remove(_childTokenId) (garment/DigitalaxGarmentNFT.sol#363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ERC721WithSameTokenURIForAllTokens.constructor(string,string).name (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#91) shadows:
- ERC721WithSameTokenURIForAllTokens.name() (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#120-122) (function)
- IERC721Metadata.name() (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#16) (function)
ERC721WithSameTokenURIForAllTokens.constructor(string,string).symbol (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#91) shadows:
- ERC721WithSameTokenURIForAllTokens.symbol() (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#127-129) (function)
- IERC721Metadata.symbol() (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#21) (function)
ERC20.constructor(string,string).name (@openzeppelin/contracts/token/ERC20/ERC20.sol#57) shadows:
- ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#66-68) (function)
ERC20.constructor(string,string).symbol (@openzeppelin/contracts/token/ERC20/ERC20.sol#57) shadows:
- ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#74-76) (function)
DigitalaxMaterials.batchCreateChildren(string[]).uri (garment/DigitalaxMaterials.sol#88) shadows:
- ERC1155.uri(uint256) (ERC1155/ERC1155.sol#70-72) (function)
- IERC1155MetadataURI.uri(uint256) (@openzeppelin/contracts/token/ERC1155/IERC1155MetadataURI.sol#20)
(function)
ERC721.constructor(string,string).name (@openzeppelin/contracts/token/ERC721/ERC721.sol#93) shadows:
- ERC721.name() (@openzeppelin/contracts/token/ERC721/ERC721.sol#122-124) (function)
- IERC721Metadata.name() (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#16) (function)
ERC721.constructor(string,string).symbol (@openzeppelin/contracts/token/ERC721/ERC721.sol#93) shadows:
- ERC721.symbol() (@openzeppelin/contracts/token/ERC721/ERC721.sol#129-131) (function)
- IERC721Metadata.symbol() (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#21) (function)
MockERC20.constructor(string,string,uint256).name (mock/MockERC20.sol#9) shadows:
- ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#66-68) (function)
MockERC20.constructor(string,string,uint256).symbol (mock/MockERC20.sol#10) shadows:
- ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#74-76) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in DigitalaxGenesisNFT.adminBuy(address) (DigitalaxGenesisNFT.sol#194-209):
External calls:
- _safeMint(_beneficiary,tokenId) (DigitalaxGenesisNFT.sol#203)
- (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
- returndata =
to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,to
kenId,_data),ERC721: transfer to non ERC721Receiver implementer) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#393-399)
External calls sending eth:
- _safeMint(_beneficiary,tokenId) (DigitalaxGenesisNFT.sol#203)
- (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
State variables written after the call(s):
- totalAdminMints = totalAdminMints.add(1) (DigitalaxGenesisNFT.sol#206)
Reentrancy in DigitalaxGarmentNFT.burn(uint256) (garment/DigitalaxGarmentNFT.sol#105-124):
External calls:
- _extractAndTransferChildrenFromParent(_tokenId,_msgSender()) (garment/DigitalaxGarmentNFT.sol#115)
- childContract.safeBatchTransferFrom(address(this),_to,_childTokenIds,_amounts,abi.encodePacked())
(garment/DigitalaxGarmentNFT.sol#346)
State variables written after the call(s):
- _burn(_tokenId) (garment/DigitalaxGarmentNFT.sol#119)
- delete _tokenURIs[tokenId] (@openzeppelin/contracts/token/ERC721/ERC721.sol#366)
- delete garmentDesigners[_tokenId] (garment/DigitalaxGarmentNFT.sol#122)
- delete primarySalePrice[_tokenId] (garment/DigitalaxGarmentNFT.sol#123)
Reentrancy in DigitalaxGarmentNFT.mint(address,string,address) (garment/DigitalaxGarmentNFT.sol#78-98):
External calls:
- _safeMint(_beneficiary,tokenId) (garment/DigitalaxGarmentNFT.sol#91)
- (success,returndata) = target.call{value: weiValue}(data)

```
(@openzeppelin/contracts/utils/Address.sol#123)
        - returndata =
to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,to
kenId,_data),ERC721: transfer to non ERC721Receiver implementer) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#441-447)
    External calls sending eth:
    - _safeMint(_beneficiary,tokenId) (garment/DigitalaxGarmentNFT.sol#91)
        - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
    State variables written after the call(s):
    - _setTokenURI(tokenId,_tokenUri) (garment/DigitalaxGarmentNFT.sol#92)
        - _tokenURIs[tokenId] = _tokenURI (@openzeppelin/contracts/token/ERC721/ERC721.sol#413)
    - garmentDesigners[tokenId] = _designer (garment/DigitalaxGarmentNFT.sol#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in DigitalaxGarmentNFT._extractAndTransferChildrenFromParent(uint256,address) (garment/
DigitalaxGarmentNFT.sol#333-349):
    External calls:
    - childContract.safeBatchTransferFrom(address(this),_to,_childTokenIds,_amounts,abi.encodePacked())
(garment/DigitalaxGarmentNFT.sol#346)
    Event emitted after the call(s):
    - TransferBatchChild(_fromTokenId,_to,address(childContract),_childTokenIds,_amounts) (garment/
DigitalaxGarmentNFT.sol#348)
Reentrancy in DigitalaxAuction._refundHighestBidder(address,uint256) (DigitalaxAuction.sol#745-758):
    External calls:
    - (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}() (DigitalaxAuction.sol#750-752)
    Event emitted after the call(s):
    - BidRefunded(_currentHighestBidder,_currentHighestBid) (DigitalaxAuction.sol#757)
Reentrancy in DigitalaxGenesisNFT.adminBuy(address) (DigitalaxGenesisNFT.sol#194-209):
    External calls:
    - _safeMint(_beneficiary,tokenId) (DigitalaxGenesisNFT.sol#203)
        - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
        - returndata =
to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,to
kenId,_data),ERC721: transfer to non ERC721Receiver implementer) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#393-399)
    External calls sending eth:
    - _safeMint(_beneficiary,tokenId) (DigitalaxGenesisNFT.sol#203)
        - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
    Event emitted after the call(s):
    - AdminGenesisMinted(_beneficiary,_msgSender(),tokenId) (DigitalaxGenesisNFT.sol#208)
Reentrancy in DigitalaxGarmentNFT.burn(uint256) (garment/DigitalaxGarmentNFT.sol#105-124):
    External calls:
    - _extractAndTransferChildrenFromParent(_tokenId,_msgSender()) (garment/DigitalaxGarmentNFT.sol#115)
        - childContract.safeBatchTransferFrom(address(this),_to,_childTokenIds,_amounts,abi.encodePacked())
(garment/DigitalaxGarmentNFT.sol#346)
    Event emitted after the call(s):
    - Approval(ownerOf(tokenId),to,tokenId) (@openzeppelin/contracts/token/ERC721/ERC721.sol#454)
        - _burn(_tokenId) (garment/DigitalaxGarmentNFT.sol#119)
    - Transfer(owner,address(0),tokenId) (@openzeppelin/contracts/token/ERC721/ERC721.sol#373)
        - _burn(_tokenId) (garment/DigitalaxGarmentNFT.sol#119)
Reentrancy in DigitalaxGenesisNFT.buy() (DigitalaxGenesisNFT.sol#118-148):
    External calls:
    - (fundsTransferSuccess) = fundsMultisig.call{value: _contributionAmount}() (DigitalaxGenesisNFT.sol#141)
    - _safeMint(_msgSender(),tokenId) (DigitalaxGenesisNFT.sol#145)
        - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
        - returndata =
to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,to
kenId,_data),ERC721: transfer to non ERC721Receiver implementer)
```

```
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#393-399)
    External calls sending eth:
    - (fundsTransferSuccess) = fundsMultisig.call{value: _contributionAmount}() (DigitalaxGenesisNFT.sol#141)
    - _safeMint(_msgSender(),tokenId) (DigitalaxGenesisNFT.sol#145)
        - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/
Address.sol#123)
    Event emitted after the call(s):
    - GenesisPurchased(_msgSender(),tokenId,_contributionAmount) (DigitalaxGenesisNFT.sol#147)
    - Transfer(address(0),to,tokenId) (ERC721/ERC721WithSameTokenURIForAllTokens.sol#321)
        - _safeMint(_msgSender(),tokenId) (DigitalaxGenesisNFT.sol#145)
Reentrancy in DigitalaxAuction.cancelAuction(uint256) (DigitalaxAuction.sol#427-463):
    External calls:
    - _refundHighestBidder(highestBid.bidder,highestBid.bid) (DigitalaxAuction.sol#453)
        - (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}()
(DigitalaxAuction.sol#750-752)
    Event emitted after the call(s):
    - AuctionCancelled(_garmentTokenId) (DigitalaxAuction.sol#462)
Reentrancy in DigitalaxGenesisNFT.increaseContribution() (DigitalaxGenesisNFT.sol#157-185):
    External calls:
    - (fundsTransferSuccess) = fundsMultisig.call{value: _amountToIncrease}() (DigitalaxGenesisNFT.sol#178)
    Event emitted after the call(s):
    - ContributionIncreased(_msgSender(),_amountToIncrease) (DigitalaxGenesisNFT.sol#184)
Reentrancy in DigitalaxGarmentFactory.mintParentWithChildren(string,address,uint256[],uint256[],address)
(garment/DigitalaxGarmentFactory.sol#72-91):
    External calls:
    - garmentTokenId = garmentToken.mint(beneficiary,garmentTokenUri,designer) (garment/
DigitalaxGarmentFactory.sol#84)
    -
materials.batchMintChildren(childTokenIds,childTokenAmounts,address(garmentToken),abi.encodePacked(gar
mentTokenId)) (garment/DigitalaxGarmentFactory.sol#87)
    Event emitted after the call(s):
    - GarmentCreated(garmentTokenId) (garment/DigitalaxGarmentFactory.sol#90)
Reentrancy in DigitalaxGarmentFactory.mintParentWithoutChildren(string,address,address) (garment/
DigitalaxGarmentFactory.sol#97-112):
    External calls:
    - garmentTokenId = garmentToken.mint(beneficiary,garmentTokenUri,designer) (garment/
DigitalaxGarmentFactory.sol#108)
    Event emitted after the call(s):
    - GarmentCreated(garmentTokenId) (garment/DigitalaxGarmentFactory.sol#111)
Reentrancy in DigitalaxAuction.placeBid(uint256) (DigitalaxAuction.sol#221-262):
    External calls:
    - _refundHighestBidder(highestBid.bidder,highestBid.bid) (DigitalaxAuction.sol#253)
        - (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}()
(DigitalaxAuction.sol#750-752)
    Event emitted after the call(s):
    - BidPlaced(_garmentTokenId,_msgSender(),bidAmount) (DigitalaxAuction.sol#261)
Reentrancy in DigitalaxAuction.resultAuction(uint256) (DigitalaxAuction.sol#315-420):
    External calls:
    - garmentNft.setPrimarySalePrice(_garmentTokenId,winningBid) (DigitalaxAuction.sol#373)
    - (platformTransferSuccess) = platformFeeRecipient.call{value: platformFeeAboveReserve}()
(DigitalaxAuction.sol#385-387)
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value:
winningBid.sub(platformFeeAboveReserve)}() (DigitalaxAuction.sol#394-396)
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value: winningBid}()
(DigitalaxAuction.sol#403-405)
    - garmentNft.safeTransferFrom(garmentNft.ownerOf(_garmentTokenId),winner,_garmentTokenId)
(DigitalaxAuction.sol#413-417)
    External calls sending eth:
    - (platformTransferSuccess) = platformFeeRecipient.call{value: platformFeeAboveReserve}()
(DigitalaxAuction.sol#385-387)
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value:
winningBid.sub(platformFeeAboveReserve)}() (DigitalaxAuction.sol#394-396)
```

```
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value: winningBid}()
(DigitalaxAuction.sol#403-405)
    Event emitted after the call(s):
    - AuctionResulted(_garmentTokenId,winner,winningBid) (DigitalaxAuction.sol#419)
Reentrancy in DigitalaxAuction.withdrawBid(uint256) (DigitalaxAuction.sol#269-302):
    External calls:
    - _refundHighestBidder(_msgSender(),previousBid) (DigitalaxAuction.sol#299)
        - (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}()
(DigitalaxAuction.sol#750-752)
    Event emitted after the call(s):
    - BidWithdrawn(_garmentTokenId,_msgSender(),previousBid) (DigitalaxAuction.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
DigitalaxAuction.placeBid(uint256) (DigitalaxAuction.sol#221-262) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_getNow() >= auction.startTime && _getNow() <=
auction.endTime,DigitalaxAuction.placeBid: Bidding outside of the auction window)
(DigitalaxAuction.sol#236-239)
DigitalaxAuction.withdrawBid(uint256) (DigitalaxAuction.sol#269-302) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_getNow() >=
highestBid.lastBidTime.add(bidWithdrawalLockTime),DigitalaxAuction.withdrawBid: Cannot withdraw until lock
time has passed) (DigitalaxAuction.sol#283-286)
    - require(bool,string)(_getNow() < auctions[_garmentTokenId].endTime,DigitalaxAuction.withdrawBid: Past
auction end) (DigitalaxAuction.sol#288-291)
DigitalaxAuction.resultAuction(uint256) (DigitalaxAuction.sol#315-420) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_getNow() > auction.endTime,DigitalaxAuction.resultAuction: The auction has not
ended) (DigitalaxAuction.sol#332-335)
DigitalaxAuction.updateAuctionEndTime(uint256,uint256) (DigitalaxAuction.sol#564-587) uses timestamp for
comparisons
    Dangerous comparisons:
    - require(bool,string)(_endTimestamp > _getNow(),DigitalaxAuction.updateAuctionEndTime: End time
passed. Nobody can bid) (DigitalaxAuction.sol#580-583)
DigitalaxAuction._createAuction(uint256,uint256,uint256,uint256) (DigitalaxAuction.sol#707-738) uses
timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_endTimestamp > _getNow(),DigitalaxAuction.createAuction: End time passed. Nobody
can bid.) (DigitalaxAuction.sol#724-727)
DigitalaxGenesisNFT.buy() (DigitalaxGenesisNFT.sol#118-148) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_getNow() >= genesisStartTimestamp && _getNow() <=
genesisEndTimestamp,DigitalaxGenesisNFT.buy: No genesis are available outside of the genesis window)
(DigitalaxGenesisNFT.sol#120-123)
DigitalaxGenesisNFT.increaseContribution() (DigitalaxGenesisNFT.sol#157-185) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(_getNow() >= genesisStartTimestamp && _getNow() <=
genesisEndTimestamp,DigitalaxGenesisNFT.increaseContribution: No increases are possible outside of the
genesis window) (DigitalaxGenesisNFT.sol#158-161)
DigitalaxGenesisNFT.updateGenesisEnd(uint256) (DigitalaxGenesisNFT.sol#214-231) uses timestamp for
comparisons
    Dangerous comparisons:
    - require(bool,string)(genesisEndTimestamp > _getNow(),DigitalaxGenesisNFT.updateGenesisEnd: End time
already passed) (DigitalaxGenesisNFT.sol#220)
DigitalaxGenesisNFT._beforeTokenTransfer(address,address,uint256) (DigitalaxGenesisNFT.sol#267-271) uses
timestamp for comparisons
    Dangerous comparisons:
    - from != address(0) && _getNow() <= genesisEndTimestamp (DigitalaxGenesisNFT.sol#268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
DigitalaxGarmentNFT._extractIncomingTokenId() (garment/DigitalaxGarmentNFT.sol#177-183) uses assembly
    - INLINE ASM (garment/DigitalaxGarmentNFT.sol#181)
```

```
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
    - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/
Address.sol#119-140) uses assembly
    - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
DigitalaxAuction.placeBid(uint256) (DigitalaxAuction.sol#221-262) compares to a boolean constant:
    -require(bool,string)(_msgSender().isContract() == false,DigitalaxAuction.placeBid: No contracts permitted)
(DigitalaxAuction.sol#227-230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used in :
    - Version used: ['0.6.12', '^0.6.0', '^0.6.2']
    - ^0.6.0 (@openzeppelin/contracts/access/AccessControl.sol#3)
    - ^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3)
    - 0.6.12 (mock/BiddingContractMock.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC1155/ERC1155Receiver.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/introspection/IERC165.sol#3)
    - ^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
    - ^0.6.2 (@openzeppelin/contracts/token/ERC1155/IERC1155MetadataURI.sol#3)
    - ^0.6.2 (@openzeppelin/contracts/token/ERC1155/IERC1155.sol#3)
    - ^0.6.0 (mock/ERC1155ReceiverMock.sol#3)
    - 0.6.12 (ERC1155/ERC1155Burnable.sol#5)
    - 0.6.12 (DigitalaxAccessControls.sol#3)
    - 0.6.12 (DigitalaxAuction.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
    - 0.6.12 (garment/DigitalaxGarmentFactory.sol#3)
    - ABIEncoderV2 (garment/DigitalaxGarmentFactory.sol#4)
    - 0.6.12 (ERC721/ERC721WithSameTokenURIForAllTokens.sol#3)
    - ^0.6.0 (ERC1155/ERC1155.sol#6)
    - ^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/utils/EnumerableMap.sol#3)
    - 0.6.12 (DigitalaxGenesisNFT.sol#3)
    - ^0.6.0 (mock/ERC1155Mock.sol#3)
    - ^0.6.0 (mock/ERC165Mock.sol#3)
    - 0.6.12 (mock/AlwaysRevertingEthReceiver.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
    - 0.6.12 (mock/MockVault.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/utils/EnumerableSet.sol#3)
    - 0.6.12 (garment/DigitalaxMaterials.sol#3)
    - ABIEncoderV2 (garment/DigitalaxMaterials.sol#4)
    - 0.6.12 (ERC998/IERC998ERC1155TopDown.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#3)
    - 0.6.12 (mock/DigitalaxGenesisNFTMock.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/math/SafeMath.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/introspection/ERC165.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
    - 0.6.12 (garment/IDigitalaxGarmentNFT.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/GSN/Context.sol#3)
    - 0.6.12 (mock/DigitalaxAuctionMock.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/utils/Strings.sol#3)
    - 0.6.12 (garment/DigitalaxGarmentNFT.sol#3)
    - 0.6.12 (mock/MockERC20.sol#3)
    - ^0.6.2 (@openzeppelin/contracts/utils/Address.sol#3)
    - ^0.6.0 (@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
    - ^0.6.0 (mock/ERC721ReceiverMock.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-
used
INFO:Detectors:
Pragma version^0.6.0 (@openzeppelin/contracts/access/AccessControl.sol#3) allows old versions
```

```
Pragma version^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3) allows old versions
Pragma version0.6.12 (mock/BiddingContractMock.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC1155/ERC1155Receiver.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/introspection/IERC165.sol#3) allows old versions
Pragma version^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721.sol#3) allows old versions
Pragma version^0.6.2 (@openzeppelin/contracts/token/ERC1155/IERC1155MetadataURI.sol#3) allows old
versions
Pragma version^0.6.2 (@openzeppelin/contracts/token/ERC1155/IERC1155.sol#3) allows old versions
Pragma version^0.6.0 (mock/ERC1155ReceiverMock.sol#3) allows old versions
Pragma version0.6.12 (ERC1155/ERC1155Burnable.sol#5) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version0.6.12 (DigitalaxAccessControls.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.11
Pragma version0.6.12 (DigitalaxAuction.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version0.6.12 (garment/DigitalaxGarmentFactory.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version0.6.12 (ERC721/ERC721WithSameTokenURIForAllTokens.sol#3) necessitates a version too recent
to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.0 (ERC1155/ERC1155.sol#6) allows old versions
Pragma version^0.6.2 (@openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/utils/EnumerableMap.sol#3) allows old versions
Pragma version0.6.12 (DigitalaxGenesisNFT.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.11
Pragma version^0.6.0 (mock/ERC1155Mock.sol#3) allows old versions
Pragma version^0.6.0 (mock/ERC165Mock.sol#3) allows old versions
Pragma version0.6.12 (mock/AlwaysRevertingEthReceiver.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#3) allows old versions
Pragma version0.6.12 (mock/MockVault.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/utils/EnumerableSet.sol#3) allows old versions
Pragma version0.6.12 (garment/DigitalaxMaterials.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version0.6.12 (ERC998/IERC998ERC1155TopDown.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#3) allows old versions
Pragma version0.6.12 (mock/DigitalaxGenesisNFTMock.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC721/ERC721.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/math/SafeMath.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/introspection/ERC165.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) allows old versions
Pragma version0.6.12 (garment/IDigitalaxGarmentNFT.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/GSN/Context.sol#3) allows old versions
Pragma version0.6.12 (mock/DigitalaxAuctionMock.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version^0.6.0 (@openzeppelin/contracts/utils/Strings.sol#3) allows old versions
Pragma version0.6.12 (garment/DigitalaxGarmentNFT.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.11
Pragma version0.6.12 (mock/MockERC20.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.11
Pragma version^0.6.2 (@openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.6.0 (@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) allows old versions
Pragma version^0.6.0 (mock/ERC721ReceiverMock.sol#3) allows old versions
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in DigitalaxAuction.resultAuction(uint256) (DigitalaxAuction.sol#315-420):
    - (platformTransferSuccess) = platformFeeRecipient.call{value: platformFeeAboveReserve}()
(DigitalaxAuction.sol#385-387)
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value:
winningBid.sub(platformFeeAboveReserve)}() (DigitalaxAuction.sol#394-396)
    - (designerTransferSuccess) = garmentNft.garmentDesigners(_garmentTokenId).call{value: winningBid}()
(DigitalaxAuction.sol#403-405)
Low level call in DigitalaxAuction._refundHighestBidder(address,uint256) (DigitalaxAuction.sol#745-758):
    - (successRefund) = _currentHighestBidder.call{value: _currentHighestBid}() (DigitalaxAuction.sol#750-752)
Low level call in DigitalaxGenesisNFT.buy() (DigitalaxGenesisNFT.sol#118-148):
    - (fundsTransferSuccess) = fundsMultisig.call{value: _contributionAmount}() (DigitalaxGenesisNFT.sol#141)
Low level call in DigitalaxGenesisNFT.increaseContribution() (DigitalaxGenesisNFT.sol#157-185):
    - (fundsTransferSuccess) = fundsMultisig.call{value: _amountToIncrease}() (DigitalaxGenesisNFT.sol#178)
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#53-59):
    - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/
Address.sol#119-140):
    - (success,returndata) = target.call{value: weiValue}(data) (@openzeppelin/contracts/utils/Address.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter BiddingContractMock.bid(uint256)._garmentTokenId (mock/BiddingContractMock.sol#14) is not in
mixedCase
Parameter DigitalaxAccessControls.hasAdminRole(address)._address (DigitalaxAccessControls.sol#59) is not in
mixedCase
Parameter DigitalaxAccessControls.hasMinterRole(address)._address (DigitalaxAccessControls.sol#68) is not in
mixedCase
Parameter DigitalaxAccessControls.hasSmartContractRole(address)._address (DigitalaxAccessControls.sol#77)
is not in mixedCase
Parameter DigitalaxAccessControls.addAdminRole(address)._address (DigitalaxAccessControls.sol#94) is not in
mixedCase
Parameter DigitalaxAccessControls.removeAdminRole(address)._address (DigitalaxAccessControls.sol#104) is
not in mixedCase
Parameter DigitalaxAccessControls.addMinterRole(address)._address (DigitalaxAccessControls.sol#114) is not in
mixedCase
Parameter DigitalaxAccessControls.removeMinterRole(address)._address (DigitalaxAccessControls.sol#124) is
not in mixedCase
Parameter DigitalaxAccessControls.addSmartContractRole(address)._address
(DigitalaxAccessControls.sol#134) is not in mixedCase
Parameter DigitalaxAccessControls.removeSmartContractRole(address)._address
(DigitalaxAccessControls.sol#144) is not in mixedCase
Parameter DigitalaxAuction.createAuction(uint256,uint256,uint256,uint256)._garmentTokenId
(DigitalaxAuction.sol#153) is not in mixedCase
Parameter DigitalaxAuction.createAuction(uint256,uint256,uint256,uint256)._reservePrice
(DigitalaxAuction.sol#154) is not in mixedCase
Parameter DigitalaxAuction.createAuction(uint256,uint256,uint256,uint256)._startTimestamp
(DigitalaxAuction.sol#155) is not in mixedCase
Parameter DigitalaxAuction.createAuction(uint256,uint256,uint256,uint256)._endTimestamp
(DigitalaxAuction.sol#156) is not in mixedCase
Parameter DigitalaxAuction.createAuctionOnBehalfOfOwner(uint256,uint256,uint256,uint256)._garmentTokenId
(DigitalaxAuction.sol#190) is not in mixedCase
Parameter DigitalaxAuction.createAuctionOnBehalfOfOwner(uint256,uint256,uint256,uint256)._reservePrice
(DigitalaxAuction.sol#191) is not in mixedCase
Parameter DigitalaxAuction.createAuctionOnBehalfOfOwner(uint256,uint256,uint256,uint256)._startTimestamp
(DigitalaxAuction.sol#192) is not in mixedCase
Parameter DigitalaxAuction.createAuctionOnBehalfOfOwner(uint256,uint256,uint256,uint256)._endTimestamp
(DigitalaxAuction.sol#193) is not in mixedCase
Parameter DigitalaxAuction.placeBid(uint256)._garmentTokenId (DigitalaxAuction.sol#221) is not in mixedCase
Parameter DigitalaxAuction.withdrawBid(uint256)._garmentTokenId (DigitalaxAuction.sol#269) is not in
mixedCase
Parameter DigitalaxAuction.resultAuction(uint256)._garmentTokenId (DigitalaxAuction.sol#315) is not in
mixedCase
```

Parameter DigitalaxAuction.cancelAuction(uint256)._garmentTokenId (DigitalaxAuction.sol#427) is not in mixedCase
Parameter DigitalaxAuction.updateMinBidIncrement(uint256)._minBidIncrement (DigitalaxAuction.sol#483) is not in mixedCase
Parameter DigitalaxAuction.updateBidWithdrawalLockTime(uint256)._bidWithdrawalLockTime (DigitalaxAuction.sol#497) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionReservePrice(uint256,uint256)._garmentTokenId (DigitalaxAuction.sol#516) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionReservePrice(uint256,uint256)._reservePrice (DigitalaxAuction.sol#517) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionStartTime(uint256,uint256)._garmentTokenId (DigitalaxAuction.sol#540) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionStartTime(uint256,uint256)._startTime (DigitalaxAuction.sol#540) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionEndTime(uint256,uint256)._garmentTokenId (DigitalaxAuction.sol#565) is not in mixedCase
Parameter DigitalaxAuction.updateAuctionEndTime(uint256,uint256)._endTimestamp (DigitalaxAuction.sol#566) is not in mixedCase
Parameter DigitalaxAuction.updateAccessControls(DigitalaxAccessControls)._accessControls (DigitalaxAuction.sol#594) is not in mixedCase
Parameter DigitalaxAuction.updatePlatformFee(uint256)._platformFee (DigitalaxAuction.sol#616) is not in mixedCase
Parameter DigitalaxAuction.updatePlatformFeeRecipient(address)._platformFeeRecipient (DigitalaxAuction.sol#631) is not in mixedCase
Parameter DigitalaxAuction.getAuction(uint256)._garmentTokenId (DigitalaxAuction.sol#656) is not in mixedCase
Parameter DigitalaxAuction.getHighestBidder(uint256)._garmentTokenId (DigitalaxAuction.sol#679) is not in mixedCase
Parameter DigitalaxGarmentFactory.createNewChild(string)._uri (garment/DigitalaxGarmentFactory.sol#47) is not in mixedCase
Parameter DigitalaxGarmentFactory.createNewChildren(string[])._uris (garment/DigitalaxGarmentFactory.sol#60) is not in mixedCase
Parameter ERC721WithSameTokenURIForAllTokens.safeTransferFrom(address,address,uint256,bytes)._data (ERC721/ERC721WithSameTokenURIForAllTokens.sol#223) is not in mixedCase
Parameter DigitalaxGenesisNFT.adminBuy(address)._beneficiary (DigitalaxGenesisNFT.sol#194) is not in mixedCase
Parameter DigitalaxGenesisNFT.updateGenesisEnd(uint256)._end (DigitalaxGenesisNFT.sol#214) is not in mixedCase
Parameter DigitalaxGenesisNFT.updateAccessControls(DigitalaxAccessControls)._accessControls (DigitalaxGenesisNFT.sol#236) is not in mixedCase
Constan DigitalaxGenesisNFT.minimumContributionAmount (DigitalaxGenesisNFT.sol#66) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DigitalaxGenesisNFT.maximumContributionAmount (DigitalaxGenesisNFT.sol#69) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DigitalaxGenesisNFT.maxGenesisContributionTokens (DigitalaxGenesisNFT.sol#78) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter MockVault.init(DigitalaxMaterials,IERC20,string)._materials (mock/MockVault.sol#14) is not in mixedCase
Parameter MockVault.init(DigitalaxMaterials,IERC20,string)._supportedERC20Asset (mock/MockVault.sol#14) is not in mixedCase
Parameter MockVault.init(DigitalaxMaterials,IERC20,string)._supportedAssetSyntheticStrandUri (mock/MockVault.sol#14) is not in mixedCase
Parameter MockVault.mintAssetBackedSyntheticMaterial(uint256)._depositAmount (mock/MockVault.sol#22) is not in mixedCase
Parameter MockVault.mintAssetBackedSyntheticMaterialToGarment(address,uint256,uint256)._garmentAddress (mock/MockVault.sol#35) is not in mixedCase
Parameter MockVault.mintAssetBackedSyntheticMaterialToGarment(address,uint256,uint256)._garmentId (mock/MockVault.sol#35) is not in mixedCase
Parameter MockVault.mintAssetBackedSyntheticMaterialToGarment(address,uint256,uint256)._depositAmount (mock/MockVault.sol#35) is not in mixedCase
Parameter MockVault.claimUnderlyingAssetFromMaterial(uint256,uint256)._strandId (mock/MockVault.sol#50) is not in mixedCase

Parameter MockVault.claimUnderlyingAssetFromMaterial(uint256,uint256)._claimAmount (mock/MockVault.sol#50) is not in mixedCase
Parameter DigitalaxMaterials.createChild(string)._uri (garment/DigitalaxMaterials.sol#57) is not in mixedCase
Parameter DigitalaxMaterials.batchCreateChildren(string[])._uris (garment/DigitalaxMaterials.sol#78) is not in mixedCase
Parameter DigitalaxMaterials.mintChild(uint256,uint256,address,bytes)._childTokenId (garment/DigitalaxMaterials.sol#109) is not in mixedCase
Parameter DigitalaxMaterials.mintChild(uint256,uint256,address,bytes)._amount (garment/DigitalaxMaterials.sol#109) is not in mixedCase
Parameter DigitalaxMaterials.mintChild(uint256,uint256,address,bytes)._beneficiary (garment/DigitalaxMaterials.sol#109) is not in mixedCase
Parameter DigitalaxMaterials.mintChild(uint256,uint256,address,bytes)._data (garment/DigitalaxMaterials.sol#109) is not in mixedCase
Parameter DigitalaxMaterials.batchMintChildren(uint256[],uint256[],address,bytes)._childTokenIds (garment/DigitalaxMaterials.sol#127) is not in mixedCase
Parameter DigitalaxMaterials.batchMintChildren(uint256[],uint256[],address,bytes)._amounts (garment/DigitalaxMaterials.sol#128) is not in mixedCase
Parameter DigitalaxMaterials.batchMintChildren(uint256[],uint256[],address,bytes)._beneficiary (garment/DigitalaxMaterials.sol#129) is not in mixedCase
Parameter DigitalaxMaterials.batchMintChildren(uint256[],uint256[],address,bytes)._data (garment/DigitalaxMaterials.sol#130) is not in mixedCase
Parameter DigitalaxMaterials.updateAccessControls(DigitalaxAccessControls)._accessControls (garment/DigitalaxMaterials.sol#152) is not in mixedCase
Parameter DigitalaxGenesisNFTMock.addContribution(uint256)._contributionAmount (mock/DigitalaxGenesisNFTMock.sol#21) is not in mixedCase
Parameter DigitalaxGenesisNFTMock.setNowOverride(uint256)._now (mock/DigitalaxGenesisNFTMock.sol#26) is not in mixedCase
Parameter DigitalaxGenesisNFTMock.setMaxGenesisContributionTokensOverride(uint256)._maxGenesisContributionTokensOverride (mock/DigitalaxGenesisNFTMock.sol#30) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (@openzeppelin/contracts/token/ERC721/ERC721.sol#245) is not in mixedCase
Parameter DigitalaxAuctionMock.setNowOverride(uint256)._now (mock/DigitalaxAuctionMock.sol#18) is not in mixedCase
Parameter DigitalaxGarmentNFT.mint(address,string,address)._beneficiary (garment/DigitalaxGarmentNFT.sol#78) is not in mixedCase
Parameter DigitalaxGarmentNFT.mint(address,string,address)._tokenUri (garment/DigitalaxGarmentNFT.sol#78) is not in mixedCase
Parameter DigitalaxGarmentNFT.mint(address,string,address)._designer (garment/DigitalaxGarmentNFT.sol#78) is not in mixedCase
Parameter DigitalaxGarmentNFT.burn(uint256)._tokenId (garment/DigitalaxGarmentNFT.sol#105) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)._operator (garment/DigitalaxGarmentNFT.sol#129) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)._from (garment/DigitalaxGarmentNFT.sol#129) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)._id (garment/DigitalaxGarmentNFT.sol#129) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)._amount (garment/DigitalaxGarmentNFT.sol#129) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes)._data (garment/DigitalaxGarmentNFT.sol#129) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._operator (garment/DigitalaxGarmentNFT.sol#154) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._from (garment/DigitalaxGarmentNFT.sol#154) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._ids (garment/DigitalaxGarmentNFT.sol#154) is not in mixedCase
Parameter DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._values (garment/DigitalaxGarmentNFT.sol#154) is not in mixedCase

```
Parameter DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._data
(garment/DigitalaxGarmentNFT.sol#154) is not in mixedCase
Parameter DigitalaxGarmentNFT.setTokenURI(uint256,string)._tokenId (garment/DigitalaxGarmentNFT.sol#213)
is not in mixedCase
Parameter DigitalaxGarmentNFT.setTokenURI(uint256,string)._tokenUri (garment/
DigitalaxGarmentNFT.sol#213) is not in mixedCase
Parameter DigitalaxGarmentNFT.setPrimarySalePrice(uint256,uint256)._tokenId (garment/
DigitalaxGarmentNFT.sol#228) is not in mixedCase
Parameter DigitalaxGarmentNFT.setPrimarySalePrice(uint256,uint256)._salePrice (garment/
DigitalaxGarmentNFT.sol#228) is not in mixedCase
Parameter DigitalaxGarmentNFT.updateAccessControls(DigitalaxAccessControls)._accessControls (garment/
DigitalaxGarmentNFT.sol#248) is not in mixedCase
Parameter DigitalaxGarmentNFT.updateMaxChildrenPerToken(uint256)._maxChildrenPerToken (garment/
DigitalaxGarmentNFT.sol#258) is not in mixedCase
Parameter DigitalaxGarmentNFT.exists(uint256)._tokenId (garment/DigitalaxGarmentNFT.sol#271) is not in
mixedCase
Parameter DigitalaxGarmentNFT.childBalance(uint256,address,uint256)._tokenId (garment/
DigitalaxGarmentNFT.sol#278) is not in mixedCase
Parameter DigitalaxGarmentNFT.childBalance(uint256,address,uint256)._childContract (garment/
DigitalaxGarmentNFT.sol#278) is not in mixedCase
Parameter DigitalaxGarmentNFT.childBalance(uint256,address,uint256)._childTokenId (garment/
DigitalaxGarmentNFT.sol#278) is not in mixedCase
Parameter DigitalaxGarmentNFT.childContractsFor(uint256)._tokenId (garment/DigitalaxGarmentNFT.sol#288)
is not in mixedCase
Parameter DigitalaxGarmentNFT.childIdsForOn(uint256,address)._tokenId (garment/
DigitalaxGarmentNFT.sol#301) is not in mixedCase
Parameter DigitalaxGarmentNFT.childIdsForOn(uint256,address)._childContract (garment/
DigitalaxGarmentNFT.sol#301) is not in mixedCase
Parameter DigitalaxGarmentNFT.totalChildrenMapped(uint256)._tokenId (garment/
DigitalaxGarmentNFT.sol#318) is not in mixedCase
Parameter DigitalaxGarmentNFT.isApproved(uint256,address)._tokenId (garment/
DigitalaxGarmentNFT.sol#325) is not in mixedCase
Parameter DigitalaxGarmentNFT.isApproved(uint256,address)._operator (garment/
DigitalaxGarmentNFT.sol#325) is not in mixedCase
Parameter MockERC20.mint(address,uint256)._to (mock/MockERC20.sol#16) is not in mixedCase
Parameter MockERC20.mint(address,uint256)._amount (mock/MockERC20.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-
conventions
INFO:Detectors:
getRoleMemberCount(bytes32) should be declared external:
    - AccessControl.getRoleMemberCount(bytes32) (@openzeppelin/contracts/access/
AccessControl.sol#95-97)
getRoleMember(bytes32,uint256) should be declared external:
    - AccessControl.getRoleMember(bytes32,uint256) (@openzeppelin/contracts/access/
AccessControl.sol#111-113)
getRoleAdmin(bytes32) should be declared external:
    - AccessControl.getRoleAdmin(bytes32) (@openzeppelin/contracts/access/AccessControl.sol#121-123)
renounceRole(bytes32,address) should be declared external:
    - AccessControl.renounceRole(bytes32,address) (@openzeppelin/contracts/access/
AccessControl.sol#170-174)
burn(address,uint256,uint256) should be declared external:
    - ERC1155Burnable.burn(address,uint256,uint256) (ERC1155/ERC1155Burnable.sol#16-23)
burnBatch(address,uint256[],uint256[]) should be declared external:
    - ERC1155Burnable.burnBatch(address,uint256[],uint256[]) (ERC1155/ERC1155Burnable.sol#25-32)
name() should be declared external:
    - ERC721WithSameTokenURIForAllTokens.name()
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#120-122)
    - ERC721.name() (@openzeppelin/contracts/token/ERC721/ERC721.sol#122-124)
symbol() should be declared external:
    - ERC721.symbol() (@openzeppelin/contracts/token/ERC721/ERC721.sol#129-131)
    - ERC721WithSameTokenURIForAllTokens.symbol()
```

```
(ERC721/ERC721WithSameTokenURIForAllTokens.sol#127-129)
tokenURI(uint256) should be declared external:
    - ERC721WithSameTokenURIForAllTokens.tokenURI(uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#134-138)
    - ERC721.tokenURI(uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#136-151)
tokenOfOwnerByIndex(address,uint256) should be declared external:
    - ERC721WithSameTokenURIForAllTokens.tokenOfOwnerByIndex(address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#143-145)
    - ERC721.tokenOfOwnerByIndex(address,uint256) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#165-167)
tokenByIndex(uint256) should be declared external:
    - ERC721WithSameTokenURIForAllTokens.tokenByIndex(uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#158-161)
    - ERC721.tokenByIndex(uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#180-183)
approve(address,uint256) should be declared external:
    - ERC721.approve(address,uint256) (@openzeppelin/contracts/token/ERC721/ERC721.sol#188-197)
    - ERC721WithSameTokenURIForAllTokens.approve(address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#166-175)
setApprovalForAll(address,bool) should be declared external:
    - ERC721WithSameTokenURIForAllTokens.setApprovalForAll(address,bool) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#189-194)
    - ERC721.setApprovalForAll(address,bool) (@openzeppelin/contracts/token/ERC721/ERC721.sol#211-216)
transferFrom(address,address,uint256) should be declared external:
    - ERC721.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#228-233)
    - ERC721WithSameTokenURIForAllTokens.transferFrom(address,address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#206-211)
safeTransferFrom(address,address,uint256) should be declared external:
    - ERC721.safeTransferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC721/
ERC721.sol#238-240)
    - ERC721WithSameTokenURIForAllTokens.safeTransferFrom(address,address,uint256) (ERC721/
ERC721WithSameTokenURIForAllTokens.sol#216-218)
balanceOf(address,uint256) should be declared external:
    - ERC1155.balanceOf(address,uint256) (ERC1155/ERC1155.sol#81-84)
balanceOfBatch(address[],uint256[]) should be declared external:
    - ERC1155.balanceOfBatch(address[],uint256[]) (ERC1155/ERC1155.sol#93-112)
setApprovalForAll(address,bool) should be declared external:
    - ERC1155.setApprovalForAll(address,bool) (ERC1155/ERC1155.sol#117-122)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
    - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (ERC1155/ERC1155.sol#134-161)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
    - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (ERC1155/ERC1155.sol#166-202)
registerInterface(bytes4) should be declared external:
    - ERC165Mock.registerInterface(bytes4) (mock/ERC165Mock.sol#8-10)
name() should be declared external:
    - ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#66-68)
symbol() should be declared external:
    - ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#74-76)
decimals() should be declared external:
    - ERC20.decimals() (@openzeppelin/contracts/token/ERC20/ERC20.sol#91-93)
totalSupply() should be declared external:
    - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#98-100)
balanceOf(address) should be declared external:
    - ERC20.balanceOf(address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#105-107)
transfer(address,uint256) should be declared external:
    - ERC20.transfer(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#117-120)
allowance(address,address) should be declared external:
    - ERC20.allowance(address,address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#125-127)
approve(address,uint256) should be declared external:
    - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#136-139)
```

```
transferFrom(address,address,uint256) should be declared external:
    - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC20/
ERC20.sol#153-157)
increaseAllowance(address,uint256) should be declared external:
    - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#171-174)
decreaseAllowance(address,uint256) should be declared external:
    - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#190-193)
baseURI() should be declared external:
    - ERC721.baseURI() (@openzeppelin/contracts/token/ERC721/ERC721.sol#158-160)
supportsInterface(bytes4) should be declared external:
    - ERC165.supportsInterface(bytes4) (@openzeppelin/contracts/introspection/ERC165.sol#35-37)
onERC1155Received(address,address,uint256,uint256,bytes) should be declared external:
    - DigitalaxGarmentNFT.onERC1155Received(address,address,uint256,uint256,bytes) (garment/
DigitalaxGarmentNFT.sol#129-149)
onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) should be declared external:
    - DigitalaxGarmentNFT.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (garment/
DigitalaxGarmentNFT.sol#154-175)
mint(address,uint256) should be declared external:
    - MockERC20.mint(address,uint256) (mock/MockERC20.sol#16-18)
onERC721Received(address,address,uint256,bytes) should be declared external:
    - ERC721ReceiverMock.onERC721Received(address,address,uint256,bytes) (mock/
ERC721ReceiverMock.sol#18-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-
declared-external
INFO:Slither:. analyzed (42 contracts with 46 detectors), 260 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

All the significant mentions of the Slither analysis have already been covered in the Manual Audit section.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Digitalax smart contracts. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Summary

Digitalax has implemented a complex smart contract logic in a very readable manner complying to most of Solidity's style guidelines. The test cases have more than 95% coverage on all significant contracts covering 796 unit test cases. It complies to Solidity best practices and follows patterns adopted by OpenZeppelin. Altogether, the code is written and demonstrates effective use of abstraction, separation of concerns, and modularity. The contracts are ready for deployment to Ethereum Mainnet.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com