PRISMA SHIELD

CollectiVerse

DEEP LOGIC AUDIT REPORT

CollectiVerse NFT Seed Sale

SEP 19 2022

PRISMA SHIELD

Table of Contents

Overview

This audit is for the CollectiVerse NFT seed sale contract, as well as the CollectiVerse Objects and Elements contracts.

Project Summary

Project Name	CollectiVerse
Network	Avalanche C-Chain
Language	Solidity

What is a Deep Logic Audit?

A deep logic smart contract audit is a human-driven code review that checks all of the code business logic for bugs, mathematical errors, and security risks. The audit verifies that the code honors the whitepaper. In addition, this service includes mainnet testing and proactive communication with the project owners to ensure full comprehension of the project to provide the best possible code review quality.

Overview

Findings Summary

		Findings Resolved
Total Findings	**8**	8
High Security Fingings	4	4
Medium Security Fingings	2	2
High Logical Findings	0	0
Medium Logical Findings	0	0
Informational Findings	2	2

ID	Section	Type	Severity	Page	Status
NFT-01	CollectiverseNFT	Security	High	8-9	Resolved
NFT-02	CollectiverseNFT	Security	High	10	Resolved
NFT-03	CollectiverseNFT	Security	High	11	Resolved
SED-01	CollectiverseSeedSale	Security	High	13	Resolved
SED-02	CollectiverseSeedSale	Security	Medium	14	Resolved
SED-03	CollectiverseSeedSale	Security	Medium	15-16	Resolved
SED-04	CollectiverseSeedSale	Logical	Informational	17	Resolved
SED-05	CollectiverseSeedSale	Logical	Informational	18	Resolved

Contract Addresses

Please ensure you are interacting with the correct contract addresses.

CollectiverseSeedSale
Caution: this contract heavily relies on off-chain data that is not audited. Please ensure trust in the team before interacting with this contract.
0x388A5b3a6220E7e88A3021cfC50c05C6C5Ea90bB

Please ensure the following in the contract:

1. Open https://snowtrace.io/address/0x388A5b3a6220E7e88A3021cfC50c05C6C5Ea90bB#readContract
2. Ensure that the owner field is 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD
3. Ensure that the wallet field is 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD

Contract Addresses

Elements CollectiverseNFT
Caution: this is an upgradeable contract. The code under the same contract address can be changed by the project owners after the audit. Please ensure trust in the team before interacting with this contract. The project owners have noted to Prisma Shield that the reason it is upgradeable is for flexibility in case new functionality is required in the future (e.g. burning or royalty information).
0x2674F116d2d2b73AEbc625ed368F378e831A3BA8

The (upgradeable) audited code is
0x2b721B6748c84AD9B58BAb5F8CbE7CcAD4e9F246

Please ensure the following in the contract:

- Open https://reqbin.com/curl, and paste the following command to the text box on the left:

curl -X POST -H 'Content-Type: application/json' --data '{"jsonrpc":"2.0", "method":"eth_getStorageAt", "params": ["0x2674F116d2d2b73AEbc625ed368F378e831A3BA8", "0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc", "latest"], "id": 43114}' https://api.avax.network/ext/bc/C/rpc

- Click Run, the result field on the right should be

0x000000000000000000000000002b721b6748c84ad9b58bab5f8cbe7ccad4e9f246

- Open https://snowtrace.io/address/0x2674F116d2d2b73AEbc625ed368F378e831A3BA8#readProxyContract
- In the getRoleMemberCount function, enter 0x0 in the role field and click Query. The result should be 1
- In the getRoleMember function, enter 0x0 in the role field and 0 in the index field and click Query. The result should be 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD
- In the getRoleMemberCount function, enter 0x97667070c54ef182b0f5858b034beac1b6f3089aa2d3188bb1e8929f4fa9b929 in the role field and click Query. The result should be 1
- In the getRoleMember function, enter 0x97667070c54ef182b0f5858b034beac1b6f3089aa2d3188bb1e8929f4fa9b929 in the role field and 0 in the index field and click Query. The result should be 0x388A5b3a6220E7e88A3021cfC50c05C6C5Ea90bB

Contract Addresses

Objects CollectiverseNFT
Caution: this is an upgradeable contract. The code under the same contract address can be changed by the project owners after the audit. Please ensure trust in the team before interacting with this contract. The project owners have noted to Prisma Shield that the reason it is upgradeable is for flexibility in case new functionality is required in the future (e.g. burning or royalty information).
0xa0074136173DfFa94bB1d7885A43Cbd291884A39

The (upgradeable) audited code is
0x2b721B6748c84AD9B58BAb5F8CbE7CcAD4e9F246

Please ensure the following in the contract:

- Open https://reqbin.com/curl, and paste the following command to the text box on the left:

curl -X POST -H 'Content-Type: application/json' --data '{"jsonrpc":"2.0", "method":"eth_getStorageAt", "params": ["0xa0074136173DfFa94bB1d7885A43Cbd291884A39", "0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc", "latest"], "id": 43114}' https://api.avax.network/ext/bc/C/rpc

- Click Run, the result field on the right should be

0x0000000000000000000000002b721b6748c84ad9b58bab5f8cbe7ccad4e9f246

- Open https://snowtrace.io/address/0xa0074136173DfFa94bB1d7885A43Cbd291884A39#readProxyContract
- In the getRoleMemberCount function, enter 0x0 in the role field and click Query. The result should be 1
- In the getRoleMember function, enter 0x0 in the role field and 0 in the index field and click Query. The result should be 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD
- In the getRoleMemberCount function, enter 0x97667070c54ef182b0f5858b034beac1b6f3089aa2d3188bb1e8929f4fa9b929 in the role field and click Query. The result should be 1
- In the getRoleMember function, enter 0x97667070c54ef182b0f5858b034beac1b6f3089aa2d3188bb1e8929f4fa9b929 in the role field and 0 in the index field and click Query. The result should be 0x388A5b3a6220E7e88A3021cfC50c05C6C5Ea90bB

Audit Findings

CollectiverseNFT

NFT-01 - Security High Severity

The use of OperatorRole makes it difficult to find out which addresses have been given that role. The operator roles have admin privileges to call the mint and setURI functions, and therefore allowing users to easily find out which addresses have been given that role is critical to allow users to make informed decisions about the security of the project.

Recommendation

Replace the usage of OperatorRole with import "@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol"; which provides greater flexibility and allows easy auditing of assigned roles:

```
import "@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol";

contract CollectiverseNFT is AccessControlEnumerableUpgradeable, ERC1155SupplyUpgradeable {
    bytes32 public constant OPERATOR_ROLE = keccak256("OPERATOR_ROLE");

    function initialize(string memory _uri) public initializer {
        __Context_init_unchained();
        __ERC165_init_unchained();
        __AccessControl_init_unchained();
        __AccessControlEnumerable_init_unchained();
        __ERC1155_init_unchained(_uri);
        __ERC1155Supply_init_unchained();
        ...
    }
```

Audit Findings

CollectiverseNFT

NFT-01 - Security High Severity

Code continued...

```
    function supportsInterface(bytes4 interfaceId)
        public
        view
        virtual
        override(AccessControlEnumerableUpgradeable, ERC1155Upgradeable)
        returns (bool)
    {
        return super.supportsInterface(interfaceId);
    }

    function mint(address _owner, uint256 _id, uint256 _amount ) external onlyRole(OPERATOR_ROLE) {
        ...
    }

    function setURI(string calldata _uri) external onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
    }
}
```

Resolution

The team has resolved this issue.

The suggested code change has been applied.

Audit Findings

CollectiverseNFT

NFT-02 - Security High Severity

The multisig gnosis address can't be used directly to deploy the contract on the blockchain, and therefore is not the contract admin on deployment. The deploying non-multisig address will need to explicitly call the grantRole function post-deployment. If the admin role is not transferred to the multisig address, the non-multisig address will have unilateral control over the contract, and will be able to call all admin functions unilaterally.

Recommendation

Call the _grantRole function with the DEFAULT_ADMIN_ROLE and the multisig address as the arguments in the initialize function to ensure that the multisig address has the admin role from deployment:

```
function initialize(string memory _uri) public initializer {
    ...
    _grantRole(
        DEFAULT_ADMIN_ROLE,
        0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD
    );
}
```

Resolution

The team has resolved this issue.

The multisig address 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD is the DEFAULT_ADMIN_ROLE on contract deployment.

Audit Findings

CollectiverseNFT

NFT-03 - Security High Severity

The setURI function is allowed to be called by operator roles, which should not be the case as that is a more severe functionality which should only be allowed to be called by the contract admin multisig address.

Recommendation

Change the setURI function to be onlyRole(DEFAULT_ADMIN_ROLE):

```
function setURI(string calldata _uri)
    external
    onlyRole(DEFAULT_ADMIN_ROLE)
{
    ...
}
```

Resolution

The team has resolved this issue.

The setURI function has been changed to only be callable by the DEFAULT_ADMIN_ROLE.

Overview

CollectiverseNFT

- This contract is used for the CollectiVerse Objects and Elements. It extends ERC1155 with the ability to mint new tokens by operator addresses (like CollectiverseSeedSale) and allows the admin roles to change the token URI. Caution: this is an upgradeable contract. The code under the same contract address can be changed by the project owners after the audit. Please ensure trust in the team before interacting with this contract. The project owners have noted to Prisma Shield that the reason it is upgradeable is for flexibility in case new functionality is required in the future (e.g. burning or royalty information).

- The getRoleMemberCount and getRoleMember publicly viewable functions contain the information about the addresses that have been assigned the DEFAULT_ADMIN_ROLE and OPERATOR_ROLE roles. getRoleMemberCount contains the number of addresses assigned to a role, and getRoleMember contains the actual addresses that have been assigned to a role.

- The mint function can be used to mint new tokens. The operator role addresses have admin powers to call this function.

- The setURI function can be used to change the token URI, which contains the off-chain metadata about the tokens (e.g. the tokens' displayable images/animations). The admin role addresses have admin powers to call this function.

Audit Findings

CollectiverseSeedSale

SED-01 - Security High Severity

The multisig gnosis address can't be used directly to deploy the contract on the blockchain, and therefore is not the contract owner on deployment. The deploying non-multisig address will need to explicitly call the transferOwnership function post-deployment. If ownership is not transferred to the multisig address, the non-multisig address will have unilateral control over the contract, and will be able to call all admin functions unilaterally.

Recommendation

Call the _transferOwnership function with the multisig address as the argument in the constructor to ensure that the multisig address is the contract owner from deployment:

```
constructor() EIP712("CollectiverseObjects", "1") {
    ...
    _transferOwnership(0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD);
}
```

Resolution

The team has resolved this issue.

The multisig address 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD is the contract owner on deployment.

Audit Findings

CollectiverseSeedSale

SED-02 - Security Medium Severity

The use of many constructor arguments make it more error-prone to audit and ensure correct deployment of the contract.

Recommendation

Remove the constructor arguments and replace them with hardcoded values in the constructor:

```
constructor() EIP712("CollectiverseObjects", "1") {
    // TODO: hardcode the addresses so that the audit is less error-prone
    elements = address(0);
    objects = address(0);
    erc20 = 0xB97EF9Ef8734C71904D8002F8b6Bc66Dd9c48a6E;
    wallet = 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD;
    signer = address(0);
    ...
}
```

Resolution

The team has resolved this issue.

The erc20 and wallet variables, which are known beforehand, have been hardcoded in the constructor. The elements, objects, and signer variables have remained as constructor arguments, and have been verified to have been set correctly.

Audit Findings

CollectiverseSeedSale

SED-03 - Security Medium Severity

The mintObject function relies on correctly signed off-chain data that is not audited. Defense-in-depth needs to be added to mintObject to reject any invalid Voucher data.

Recommendation

Add require statements in mintObject to validate Voucher data. For example:

- Check that _voucher.preminedId is in the _voucher.elementIds array
- Check that _voucher.price is not zero
- Check that there are no repeated fields in the _voucher.elementIds array
- Check that there are no zeros in the _voucher.elementAmounts array

Example of the first bullet point:

```
function mintObject(address _owner, Voucher calldata _voucher) external returns (uint256) {
    ...
    bool preInElements = false;
    for (uint256 i = 0; i < _voucher.elementIds.length; i++) {
        minable[_id][_voucher.elementIds[i]] = _voucher.elementAmounts[i];
        if (_voucher.elementIds[i] == _pre) {
            preInElements = true;
        }
    }
    require(
        preInElements,
        "Incorrectly signed Voucher, preminedId was not found in elementIds."
    );
    ...
}
```

Audit Findings

CollectiverseSeedSale

SED-03 - Security Medium Severity

Resolution

The team has resolved this issue.

The team has added a check that _voucher.price is within a specified allowed range that can be changed but defaults to 50-2000 USDC.

Audit Findings

CollectiverseSeedSale

SED-04 - Logical Informational Severity

The erc20, elements, and objects variables are only set in the constructor, and can therefore be changed to be immutable.

Recommendation

Change the erc20, elements, and objects variables to immutable:

```
address public immutable erc20;
address public immutable elements;
address public immutable objects;
```

Resolution

The team has resolved this issue.

The erc20, elements, and objects variables have been changed to be immutable.

Audit Findings

CollectiverseSeedSale

SED-05 - Logical Informational Severity

The `using ECDSA for bytes32;` statement is unused and can be removed.

Recommendation

Delete the `using ECDSA for bytes32;` statement.

Resolution

The unused `using ECDSA for bytes32;` statement has ben deleted.

Overview

CollectiverseSeedSale

- This contract provides the functionality to allow users to buy CollectiVerse Objects and Elements ERC1155 tokens using their USDC tokens. Caution: this contract heavily relies on off-chain data that is not audited. Please ensure trust in the team before interacting with this contract.

- The owner publicly viewable variable is the contract owner, and is expected to be the 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD multisig address. The contract owner has admin powers to change this address.

- The signer publicly viewable variable is the address used to sign the off-chain data that is used to determine the CollectiVerse Objects and Elements information and their price. The contract owner has admin powers to change this address.

- The wallet publicly viewable variable is the address that receives the USDC tokens users use to by the CollectiVerse Objects and Elements. It defaults to the 0x233E70CdE8FeAb985CCAA5938C579cc47B1ffAeD multisig address. The contract owner has admin powers to change this address.

- The minimumPrice and maximumPrice publicly viewable variables dictate the minimum and maximum allowed USDC prices of a single CollectiVerse Object. These default to 50 USDC and 2000 USDC respectively. The contract owner has admin powers to change these values.

- The minable publicly viewable variable contains the information about the remaining CollectiVerse Elements that can still be minted. This information is meant to be used in future smart contracts.

Overview

CollectiverseSeedSale

- The whitelist and whitelistPurchased publicly viewable variables contain the information about all the whitelist addresses. whitelist contains the maximum number of CollectiVerse Objects that can be minted by each address through the whitelist, and whitelistPurchased contains the current number of CollectiVerse Objects that have been minted by each address through the whitelist. A whitelisted address is only allowed to mint through the whitelist the maximum number of CollectiVerse Objects assigned to it in the whitelist variable.

- The mintObject function can be used by users to buy CollectiVerse Objects and Elements by paying USDC. The ids and amounts of each are dictated by the _voucher argument, which comes from off-chain data created by the project owners.

- The setSettings function can be used to change the signer and wallet addresses and the minimumPrice and maximumPrice values. The contract owner has admin powers to call this function.

- The whitelistAddresses function can be used to set the whitelist variable to change the maximum number of purchasable Collectiverse Objects of any address through the whitelist. The contract owner has admin powers to call this function.

How to Interpret Findings

Security - High Severity

Indicates that users' funds are at risk or that there is a high probability of exploitation.

Security - Medium Severity

No risk to the protocol or those who interact with it, however it should be highlighted nonetheless.

Logical - High Severity

Indicates that the errors puts users' funds at risk, or can result in significant functional failure in the code.

Logical - Medium Severity

Indicates some functional failure or discrepancy in the code.

Logical - Informational

Minor discrepancy between the intended functionality of the code and the implementation, which does not result in functional failure, or a recommendation to improve the logic.

Yellow Text

Indicates centralization of control and admin powers.

Disclaimer

The information in this deep logic audit report objectively describes the smart contracts being audited, and points out logical and mathematical errors, security risks and vulnerabilities, and optimization opportunities in the audited code. This deep logic audit does not ensure the correctness or authenticity of any software or dApp that interacts with or claims to interact with any smart contract.

This audit report does not constitute any advice whatsoever. You are solely responsible for conducting your own due diligence and consulting your financial advisor before making any investment decisions. While our deep logic audits raise the level of security, reliability, mathematical accuracy, and logical soundness of the smart contracts reviewed, they do not amount to any form of warranty or guarantee that the reviewed smart contracts are void of any weaknesses, vulnerabilities, or bugs. Prisma Shield and its founders, employees, owners, and associates are not liable for any damage or loss of funds. Please ensure trust in the team prior to investing as this deep logic audit does not guarantee the promised use of your funds.

You can find the full disclaimer, terms and conditions, and privacy policy on the prismashield.com website.

PRISMA SHIELD

Introducing Deep Logic Smart Contract Auditing to Web3

prismashield.com

prismashield@gmail.com

PrismaShield

PrismaShield