Dec 24, 2024

ZAN

Security Assessment

Rare Shop

—

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

Rare Shop is a project that allows users to purchase NFTs using USDC and USDT, and exercise those NFTs to acquire real-world assets. This report has been prepared for Rare Shop project to discover issues and vulnerabilities in the source code of this project as well as any contract dependencies that were not part of an officially recognized library.

Conducted by Static Analysis, Formal Verification and Manual Review, we have identified **1 High, 2 Low and 9 Informational issues** in commit 75bfbc7.

The project team **has resolved all the security vulnerabilities and informational issues except for the issues described in I-08 and I-09** in commit a9914b0.

1.2. Project Summary

Project Name	Rare Shop
Platform	Mint
Language	Solidity
Codebase	Audit 1: • https://github.com/Rare-Shop/rareshop-contracts/tree/75bfbc75bfd6fc32f8fdd16a5bf4bdd1a06dfc1f Final Audit: • https://github.com/Rare-Shop/rareshop-contracts/tree/a9914b0296840ae7a11619e8a78a5ba27fdb18c9

1.3. Assessment Summary

Delivery Date	Dec 24, 2024
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File	File Hash
1	/rareshop/src/templates/RareshopSKUContract.sol	ba003ad35b844d1a2899787b31270849
2	/rareshop/src/RareshopPlatformContract.sol	21016ede1547dc34ce9983940a8457e
3	/rareshop/src/interfaces/IERC7765.sol	cec8cacefb758ea45b17197fa7076fd6
4	/rareshop/src/templates/RareshopBrandContract.sol	ca7e82f318b54c3b9daf8db36d99856c

ID	File	File Hash
5	/rareshop/src/interfaces/IERC7765Metadata.sol	8a3501d37bfad2eda42250d41c3b34

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

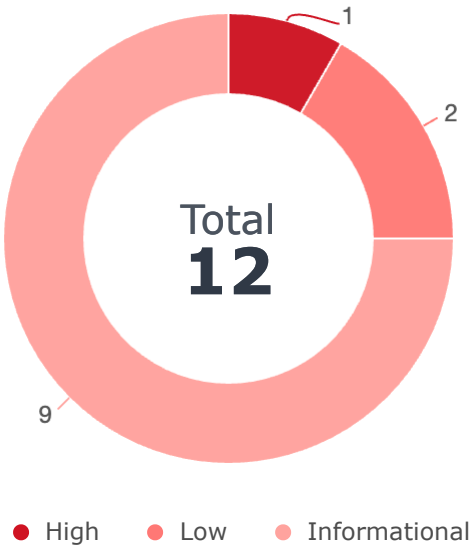
2.2. Optimization Suggestion

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings



Code Security

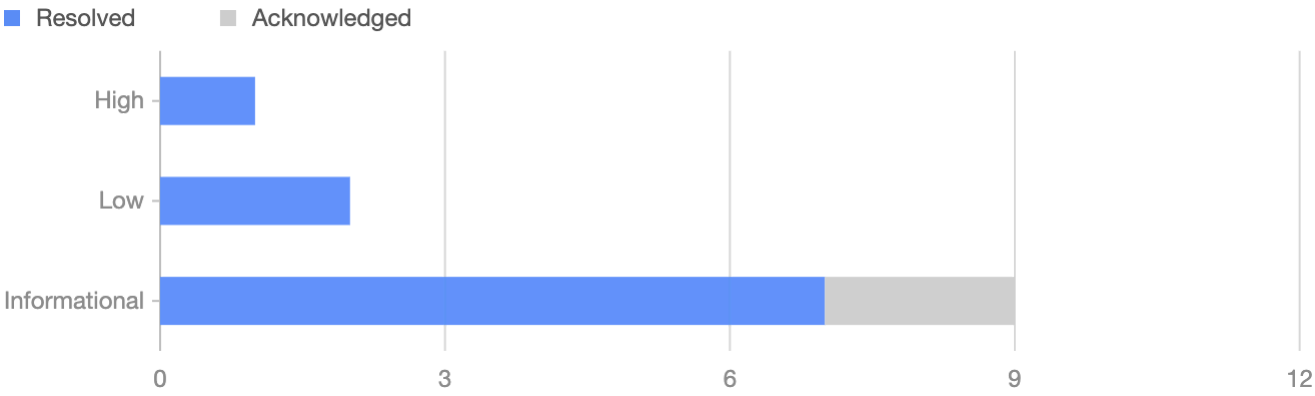
Low: 2

Optimization Suggestion

Informational: 9

Business Security

High: 1



ID	Title	Category	Severity	Status
H-01	Unlimited Minting	Business Security	High	Resolved
L-02	Use _safeMint Instead of _mint for ERC721	Code Security	Low	Resolved
L-03	Use Unsafe ABI Calls	Code Security	Low	Resolved
I-04	Improve the Inspection of SKU Config	Optimization Suggestion	Informational	Resolved
I-05	Eliminate Redundant Code	Optimization Suggestion	Informational	Resolved
I-06	Optimizable Verification of Merkle Tree Proofs	Optimization Suggestion	Informational	Resolved
I-07	Optimizable Conversion of Hexadecimal Strings	Optimization Suggestion	Informational	Resolved
I-08	Set the Constant to Private	Optimization Suggestion	Informational	Acknowledged
I-09	Redundant Getter Function	Optimization Suggestion	Informational	Acknowledged

ID	Title	Category	Severity	Status
I-10	Cache State Variables That Are Read Multiple Times within a Function	Optimization Suggestion	Informational	Resolved
I-11	Floating Pragma	Optimization Suggestion	Informational	Resolved
I-12	Test Cases Fail to Encompass All Essential Functionalities	Optimization Suggestion	Informational	Resolved

H-01: Unlimited Minting

High: Business Security

File Location: /rareshop/src/templates/RareshopSKUContract.sol:173

Description

The state variable `minted` is used to record the current number of NFTs that have been minted. In the `mint` function, the `minted` variable is used to determine whether the next minting will exceed the maximum supply limit. However, the `minted` variable has never been updated, resulting in the maximum supply constraint becoming ineffective, allowing users to mint NFTs without limit.

/rareshop/src/templates/RareshopSKUContract.sol

```
161     function mint(
162         address _payTokenAddress,
163         uint256 _amounts,
164         bytes32[] calldata _whiteListProof
165         )
166         external
167         checkWhiteList(_whiteListProof)
168         returns(uint256[] memory)
169     {
170         require(mintable, "mint not available");
171         require(_payTokenAddress == USDT_ADDRESS || _payTokenAddress ==
            USDC_ADDRESS, "Only supporting USDT/USDC");
172         require(block.timestamp >= config.startTime && block.timestamp
            <= config.endTime, "Out of sell time range");
173         require((minted + _amounts) <= config.supply, "mint amounts
            exceed supply");
174
175         address sender = _msgSender();
176         require((mintAmounts[sender] + _amounts) <= config.userLimit,
            "user mint amounts exceed limit");
```

Recommendation

It is recommended to update the `minted` variable promptly in the `mint` function.

Alleviation

Resolved in commit a9914b0.

L-02: Use _safeMint Instead of _mint for ERC721

Low: Code Security

File Location: /rareshop/src/templates/RareshopSKUContract.sol:189

Description

It is recommended to use OpenZeppelin's `_safeMint` instead of the `_mint` function to mint ERC721 tokens, as the former ensures that the recipient is an EOA or a contract that implements the `onERC721Received` callback function. This helps to avoid the scenario where the minted ERC721 tokens are stuck in a particular address.

/rareshop/src/templates/RareshopSKUContract.sol

```
188   for (uint256 i = 0; i < _amounts;) {
189       _mint(sender, nextTokenId);
190       mintedTokenIds[i] = nextTokenId++;
191       unchecked {
192           ++i;
193       }
194   }
```

Recommendation

It is recommended to use OpenZeppelin's `_safeMint` instead of the `_mint` function to mint ERC721 tokens.

Alleviation

Resolved in commit a9914b0.

L-03: Use Unsafe ABI Calls

Low: Code Security

File Location:
/rareshop/src/RareshopPlatformContract.sol:49
/rareshop/src/templates/RareshopBrandContract.sol:72

Description

The abi.encodeCall() function was introduced in Solidity version 0.8.0 and provides a type-safe way to encode function calls. Unlike abi.encodeSignature() and abi.encodeWithSelector(), which are not type-safe, abi.encodeCall() verifies that the function signature matches the provided arguments.

/rareshop/src/RareshopPlatformContract.sol

```
47          address brandCollection = Clones.cloneDeterministic
            (brandImplementationTypes[_collectionType], salt);
48
49          (bool success, bytes memory returnData) = brandCollection.call
            (abi.encodeWithSelector(
50              BRAND_INIT_SELECTOR, msg.sender, _name, _extendData));
51          if (!success) {
```

/rareshop/src/templates/RareshopBrandContract.sol

```
70              keccak256(abi.encode(msg.sender, _name, block.timestamp))
71          );
72          (bool success, bytes memory returnData) = skuCollection.call(abi.
            encodeWithSelector(
73              SKU_INIT_SELECTOR, address(this), _name, _symbol,
                _skuConfigData, _privilegeData));
74          if (!success) {
```

Recommendation

Use abi.encodeCall() instead of abi.encodeSignature()/abi.encodeWithSelector()

Alleviation

Resolved in commit a9914b0.

I-04: Improve the Inspection of SKU Config

Informational: Optimization Suggestion

File Location: /rareshop/src/templates/RareshopSKUContract.sol:132,133,318,319

Description

It is recommended to improve the checks for the `config` in the `__SKUConfig_init` and `updateSKUConfig` functions, such as ensuring that `config.endTime` is greater than the current timestamp, and that `config.userLimit` is less than `config.supply`.

/rareshop/src/templates/RareshopSKUContract.sol

```
127     function __SKUConfig_init(bytes calldata _configData) internal {
128         require(_configData.length > 0, "_configData can not be empty");
129
130         config = abi.decode(_configData, (SKUConfig));
131         require(config.supply > 0, "supply must be larger than 0");
132         require(config.startTime < config.endTime, "startTime must be
            smaller than endTime");
133         require(config.userLimit > 0, "userLimit must be larger than 0");
134         require(config.paymentReceipientAddress != address(0),
            "paymentReceipientAddress can not be empty");
135     }
```

/rareshop/src/templates/RareshopSKUContract.sol

```
308     function updateSKUConfig(
309         uint64 _supply,
310         uint64 _mintPrice,
311         uint64 _userLimit,
312         uint64 _startTime,
313         uint64 _endTime,
314         address _paymentReceipientAddress,
315         bytes32 _whiteListRoot
316     ) external onlyAdmin {
317         require(_supply > 0, "supply must be larger than 0");
318         require(_startTime < _endTime, "startTime must be smaller than
            endTime");
319         require(_userLimit > 0, "userLimit must be larger than 0");
320         require(_paymentReceipientAddress != address(0),
            "paymentReceipientAddress can not be empty");
321
```

Recommendation

It is recommended to improve the `config` check according to the description above.

Alleviation

Resolved in commit a9914b0.

I-05: Eliminate Redundant Code

Informational: Optimization Suggestion

File Location: /rareshop/src/templates/RareshopSKUContract.sol:18,417

Description

The `RareshopSKUContract` contract inherits from the `OwnableUpgradeable` contract but does not utilize any of its access control functions. Additionally, the `mockConfigData` function is solely intended for debugging purposes. Removing this unused code will improve maintainability, enhance readability, optimize performance, bolster security, and reduce the risks inherent in software development.

/rareshop/src/templates/RareshopSKUContract.sol

```
16  contract RareshopSKUContract is
17      Initializable,
18      OwnableUpgradeable,
19      ERC721Upgradeable,
20      IERC7765,
21      IERC7765Metadata
22  {
```

/rareshop/src/templates/RareshopSKUContract.sol

```
417     function mockConfigData(
418         SKUConfig memory _config,
419         Privilege[] memory _privileges
420     ) external pure returns (bytes memory, bytes memory) {
421         return (abi.encode(_config), abi.encode(_privileges)); // for
            debugging
422     }
```

Recommendation

Refrain from inheriting `OwnableUpgradeable` in the `RareshopSKUContract` and remove the debugging `mockConfigData` function.

Alleviation

Resolved in commit a9914b0.

I-06: Optimizable Verification of Merkle Tree Proofs

Informational: Optimization Suggestion

File Location: /rareshop/src/templates/RareshopSKUContract.sol:91

Description

The `checkWhiteList` modifier is a custom implementation designed to verify Merkle Tree proofs. However, it could be optimized by directly invoking the `verify` function from OpenZeppelin's `MerkleProof` contract. Leveraging existing library code, rather than developing new code from the ground up, offers substantial advantages such as time efficiency, superior quality, simplified maintenance, improved collaboration, and alignment with best practices in software development.

/rareshop/src/templates/RareshopSKUContract.sol

```
91      modifier checkWhiteList(bytes32[] calldata proof) {
92          if(config.whiteListRoot != 0){
93              bytes32 computedHash = keccak256(abi.encode(_msgSender()));
94              for (uint256 i = 0; i < proof.length;) {
95                  if(uint256(computedHash) < uint256(proof[i])) {
96                      computedHash = keccak256(abi.encode(computedHash,
                        proof[i]));
97                  } else {
98                      computedHash = keccak256(abi.encode(proof[i],
                        computedHash));
99                  }
100                 unchecked {
101                     ++i;
102                 }
103             }
104             require(computedHash == config.whiteListRoot, "msgSender not
                in whitelist");
105         }
106         _;
107     }
```

Recommendation

Utilize the `verify` function from OpenZeppelin's `MerkleProof` contract to verify Merkle Tree proofs.

Alleviation

Resolved in commit a9914b0.

I-07: Optimizable Conversion of Hexadecimal Strings

Informational: Optimization Suggestion

File Location: /rareshop/src/templates/RareshopSKUContract.sol:391

Description

The `toAsciiString` method is a custom implementation designed to convert an address into its ASCII string hexadecimal representation. However, it could be optimized by directly invoking the `toHexString` function from OpenZeppelin's `Strings` contract. Leveraging existing library code, rather than developing new code from the ground up, offers substantial advantages such as time efficiency, superior quality, simplified maintenance, improved collaboration, and alignment with best practices in software development.

/rareshop/src/templates/RareshopSKUContract.sol

```
391     function toAsciiString(address x) internal pure returns (string
        memory) {
392         bytes memory s = new bytes(40);
393         for (uint i = 0; i < 20;) {
394             bytes1 b = bytes1(uint8(uint(uint160(x)) / (2**(8*(19 -
                i)))));
395             bytes1 hi = bytes1(uint8(b) / 16);
396             bytes1 lo = bytes1(uint8(b) - 16 * uint8(hi));
```

Recommendation

Utilize the `toHexString` function from OpenZeppelin's `Strings` contract to convert an address into its ASCII string hexadecimal representation.

Alleviation

Resolved in commit a9914b0.

I-08: Set the Constant to Private

Informational: Optimization Suggestion

File Location:
/rareshop/src/templates/RareshopBrandContract.sol:10
/rareshop/src/templates/RareshopSKUContract.sol:55,56

Description

For constants, if the visibility is set to public, the compiler will automatically generate a getter function for it, which will consume more gas during deployment.

/rareshop/src/templates/RareshopBrandContract.sol

```
 8  contract RareshopBrandContract is OwnableUpgradeable {
 9
10      string public constant SKU_BASE_URL = "https://images.rare.shop/";
11
12      bytes4 private constant SKU_INIT_SELECTOR =
```

/rareshop/src/templates/RareshopSKUContract.sol

```
53      );
54
55      address public constant USDT_ADDRESS =
        0xED85184DC4BECf731358B2C63DE971856623e056;
56      address public constant USDC_ADDRESS =
        0xBAfC2b82E53555ae74E1972f3F25D8a0Fc4C3682;
57
```

/rareshop/src/templates/RareshopSKUContract.sol

```
54
55      address public constant USDT_ADDRESS =
        0xED85184DC4BECf731358B2C63DE971856623e056;
56      address public constant USDC_ADDRESS =
        0xBAfC2b82E53555ae74E1972f3F25D8a0Fc4C3682;
57
58      uint256 public nextTokenId;
```

Recommendation

It is recommended to set the visibility of constants to private instead of public.

Alleviation

The project team acknowledged the issue and decided to keep no change.

I-09: Redundant Getter Function

Informational: Optimization Suggestion

File Location: /rareshop/src/RareshopPlatformContract.sol:74

Description

A state variable with public visibility comes with a getter function to obtain the value of the variable, so there is no need to explicitly define a function to obtain the value of the variable. When deploying a contract, remove extra functions will save gas. About 37000 gas can be saved with optimization turned off while 6800 gas can be saved vice versa.

/rareshop/src/RareshopPlatformContract.sol

```
72      }
73
74      function getBrandContracts() external view returns (address[] memory)
        {
75          return brandContracts;
76      }
```

Recommendation

It is recommended to remove the explicitly defined function or change the visibility of the state variable to private or internal.

Alleviation

The project team acknowledged the issue and decided to keep no change.

I-10: Cache State Variables That Are Read Multiple Times within a Function

Informational: Optimization Suggestion

File Location:
/rareshop/src/RareshopPlatformContract.sol:44
/rareshop/src/templates/RareshopBrandContract.sol:81
/rareshop/src/templates/RareshopSKUContract.sol:176

Description

When a state variable is read multiple times in a function, using a local variable to cache the state variable can avoid frequently reading data from storage, thereby saving gas.

/rareshop/src/RareshopPlatformContract.sol

```
42          returns (address)
43      {
44          require(brandImplementationTypes[_collectionType] != address(0),
            "Invalid Implementation Type");
45
46          bytes32 salt = keccak256(abi.encode(msg.sender, _name, block.
            timestamp));
```

/rareshop/src/templates/RareshopBrandContract.sol

```
79
80          skuContracts[nextSKUId] = skuCollection;
81          skuContractIds[skuCollection] = nextSKUId;
82
83          emit RareshopSKUCreated(msg.sender, skuCollection, nextSKUId,
            _name);
```

/rareshop/src/templates/RareshopSKUContract.sol

```
174
175         address sender = _msgSender();
176         require((mintAmounts[sender] + _amounts) <= config.userLimit,
            "user mint amounts exceed limit");
177
178         IERC20 erc20Token = IERC20(_payTokenAddress);
```

Recommendation

When a state variable is read multiple times in a function, it is recommended to use a local variable to cache the state variable.

Alleviation

Resolved in commit a9914b0.

I-11: Floating Pragma

Informational: Optimization Suggestion

File Location:
/rareshop/src/RareshopPlatformContract.sol /rareshop/src/templates/RareshopBrandContract.sol:2
/rareshop/src/interfaces/IERC7765.sol:2
/rareshop/src/interfaces/IERC7765Metadata.sol:2
/rareshop/src/templates/RareshopSKUContract.sol:2

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

/rareshop/src/RareshopPlatformContract.sol
/rareshop/src/templates/RareshopBrandContract.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.20;
3
4  import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.
   sol";
```

/rareshop/src/interfaces/IERC7765.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity >=0.7.0 <0.9.0;
3
4  /// @title ERC-7765 Privileged Non-Fungible Tokens Tied To Real World
   Assets
```

/rareshop/src/interfaces/IERC7765Metadata.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity >=0.7.0 <0.9.0;
3
4  /// @title ERC-7765 Privileged Non-Fungible Tokens Tied To Real World
   Assets, optional metadata extension
```

/rareshop/src/templates/RareshopSKUContract.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.20;
3
4  import "@openzeppelin/contracts-upgradeable/token/ERC721/
   ERC721Upgradeable.sol";
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

Resolved in commit a9914b0.

I-12: Test Cases Fail to Encompass All Essential Functionalities

Informational: Optimization Suggestion

File Location:

Description

The test cases neglect to cover crucial contract functionalities, such as minting privileged non-fungible tokens and exercising a specific privilege of a token. Failing to address these key aspects in unit tests constitutes a major oversight in contract development, potentially leading to more bugs, higher maintenance costs, and a diminished overall quality of the project.

Recommendation

Enhance the test cases to ensure they cover all critical functionalities, thereby guaranteeing that each component of the contract operates correctly and contributes to the project's overall reliability and maintainability.

Alleviation

Resolved in commit a9914b0.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

Contract	FuncName	Visibility	Mutability	Modifiers
RareshopPlatformContract	_CTOR_	public	N	
RareshopPlatformContract	initialize	external	Y	initializer
RareshopPlatformContract	createBrandCollection	external	Y	
RareshopPlatformContract	setBrandImplementationTypes	external	Y	onlyOwner
RareshopPlatformContract	setSKUImplementationTypes	external	Y	onlyOwner
RareshopPlatformContract	getBrandContracts	external	N	
RareshopPlatformContract	_authorizeUpgrade	internal	N	onlyOwner
RareshopBrandContract	_CTOR_	public	N	
RareshopBrandContract	initialize	external	Y	initializer
RareshopBrandContract	createSKUCollection	external	Y	onlyAdmin
RareshopBrandContract	addAdmin	external	Y	onlyOwner
RareshopBrandContract	removeAdmin	external	Y	onlyOwner
RareshopBrandContract	isAdmin	external	N	
RareshopBrandContract	getSKUAddresses	external	N	
RareshopSKUContract	_CTOR_	public	N	

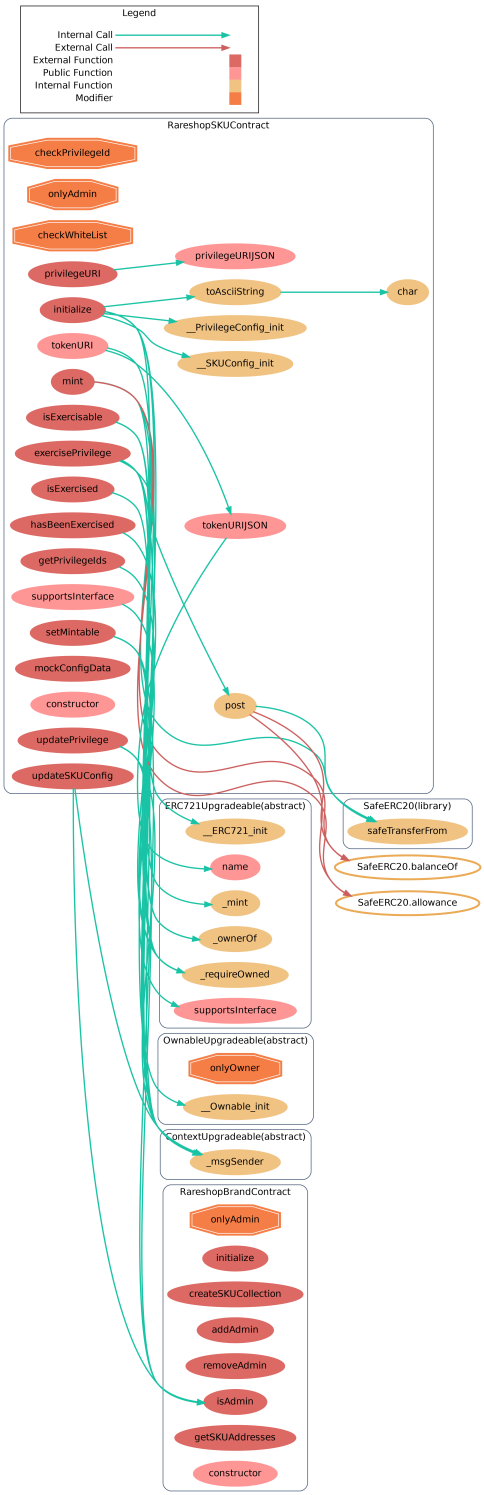
Contract	FuncName	Visibility	Mutability	Modifiers
RareshopSKUContract	initialize	external	Y	initializer
RareshopSKUContract	__SKUConfig_init	internal	N	
RareshopSKUContract	__PrivilegeConfig_init	internal	N	
RareshopSKUContract	mint	external	Y	checkWhiteList
RareshopSKUContract	exercisePrivilege	external	Y	checkPrivilegeId
RareshopSKUContract	post	internal	N	
RareshopSKUContract	isExercisable	external	N	checkPrivilegeId
RareshopSKUContract	isExercised	external	N	checkPrivilegeId
RareshopSKUContract	hasBeenExercised	external	N	checkPrivilegeId
RareshopSKUContract	getPrivilegeIds	external	N	
RareshopSKUContract	updatePrivilege	external	Y	checkPrivilegeId, onlyAdmin
RareshopSKUContract	updateSKUConfig	external	Y	onlyAdmin
RareshopSKUContract	setMintable	external	Y	onlyAdmin
RareshopSKUContract	tokenURI	public	N	
RareshopSKUContract	tokenURIJSON	public	N	
RareshopSKUContract	privilegeURI	external	N	checkPrivilegeId

Contract	FuncName	Visibility	Mutability	Modifiers
RareshopSKUContract	privilegeURIJSON	public	N	
RareshopSKUContract	toAsciiString	internal	N	
RareshopSKUContract	char	internal	N	
RareshopSKUContract	supportsInterface	public	N	
RareshopSKUContract	mockConfigData	external	N	

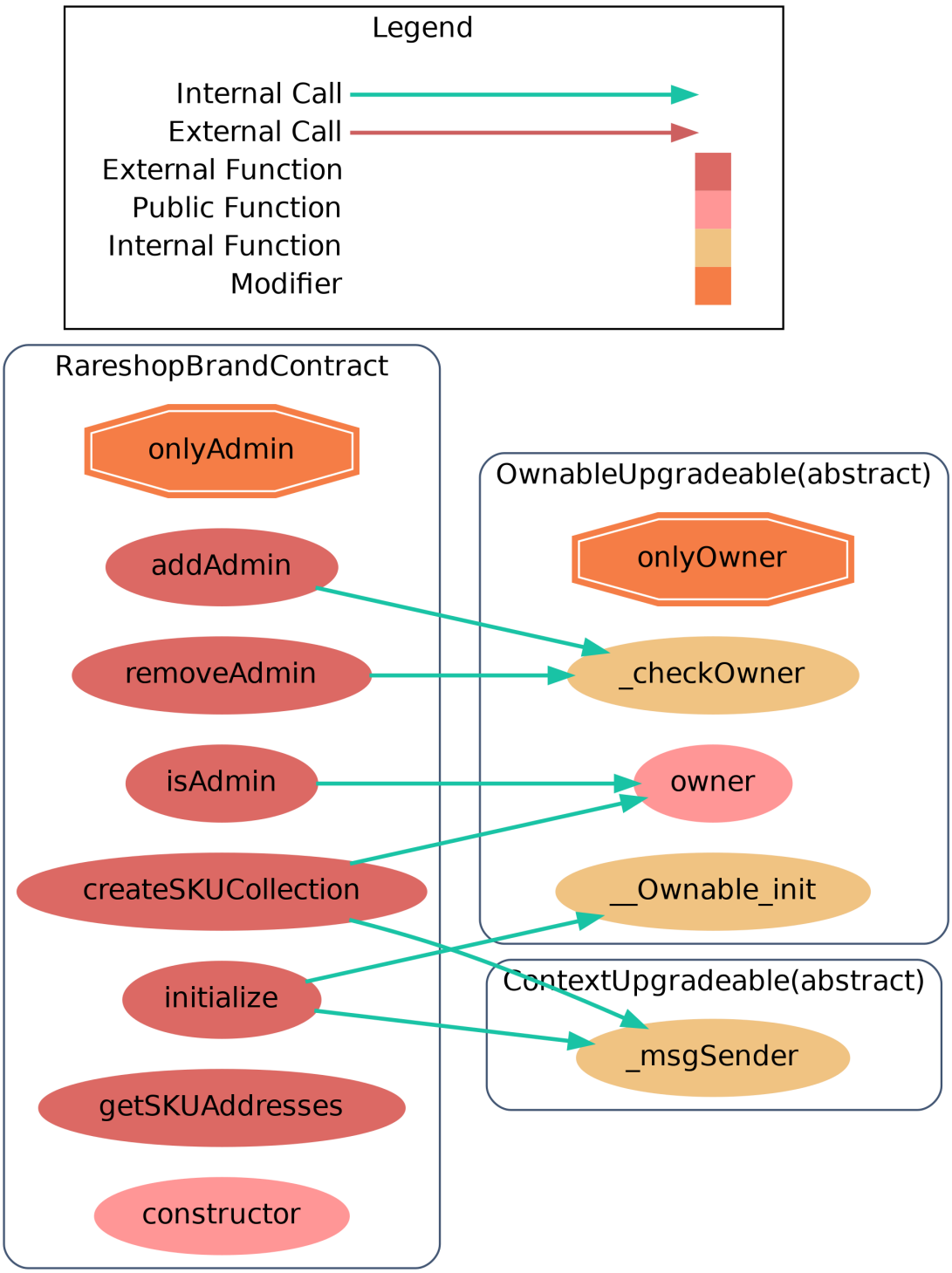
5. Appendix

5.2 Call Graph

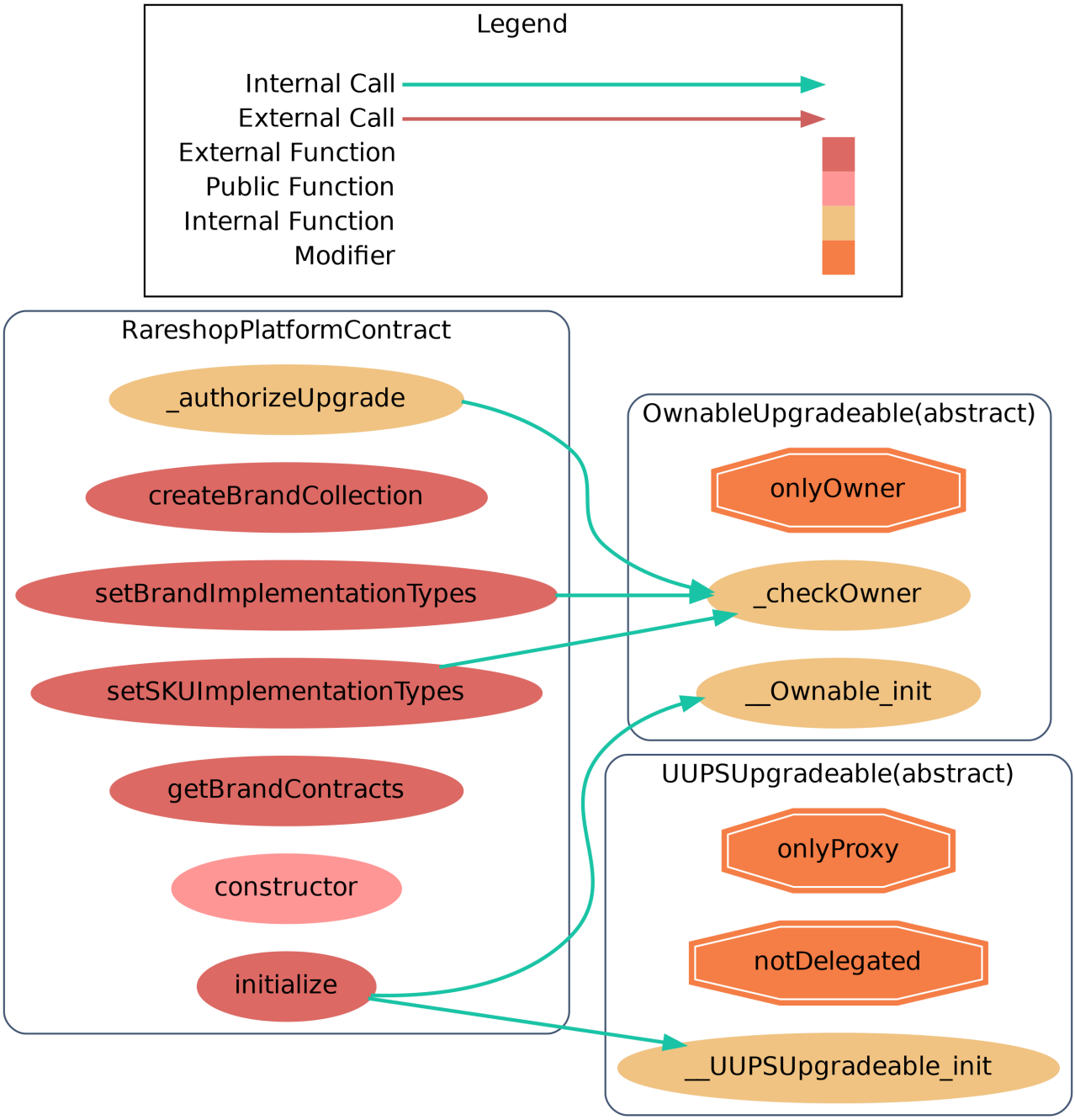
RareshopSKUContract



RareshopBrandContract



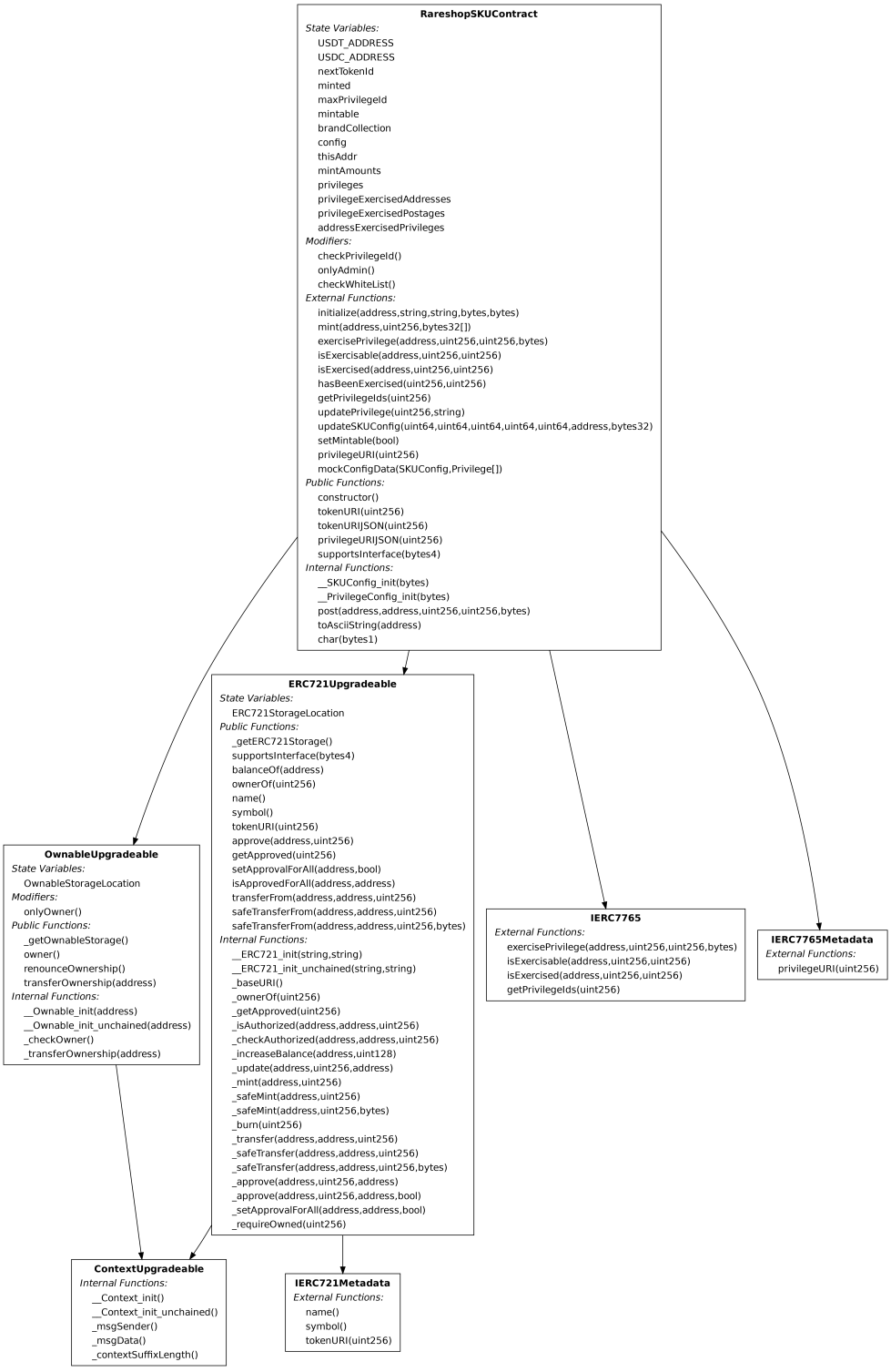
RareshopPlatformContract



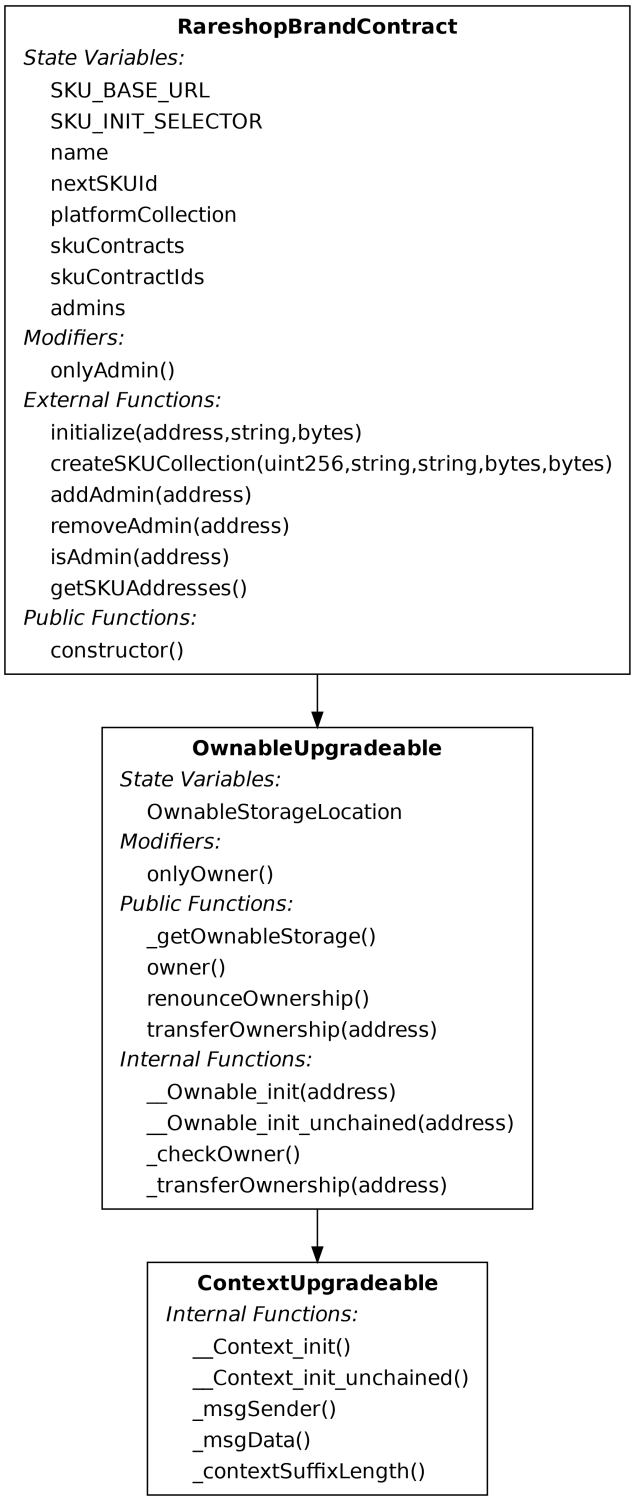
5. Appendix

5.3 Inheritance Graph

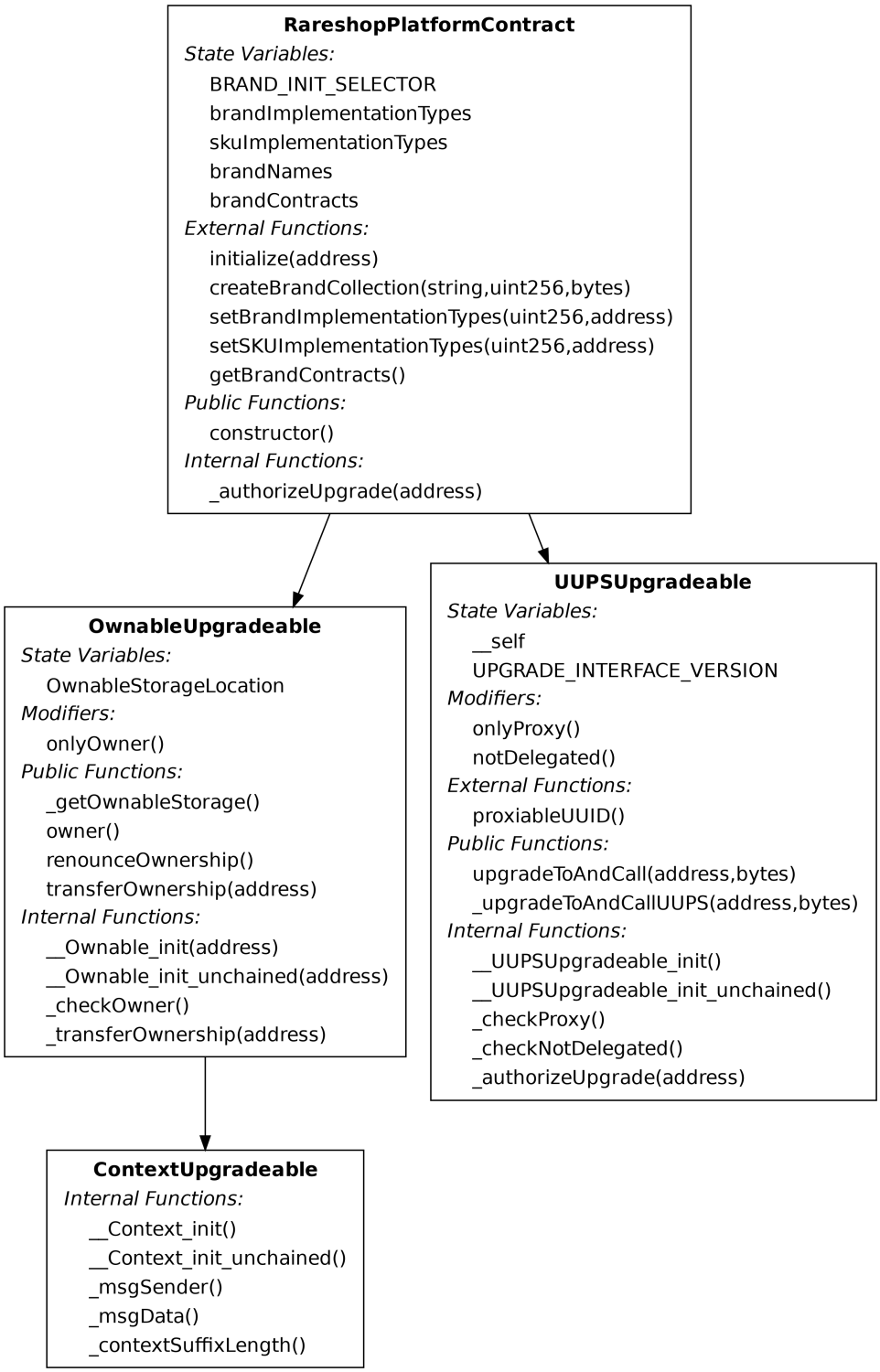
RareshopSKUContract



RareshopBrandContract



RareshopPlatformContract



5.4 Formal Verification Metadata

1. Upon the successful execution of the function `mint`, the contract should be in a mintable state.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`mint`, the contract should be in a mintable state."} old(mintable) == true;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

2. Should the function `mint` execute without fail, the pay token address must be either USDT or USDC.

```
/// #if_succeeds {:msg "Should the function `mint` execute without fail,
the pay token address must be either USDT or USDC."} _payTokenAddress ==
USDT_ADDRESS || _payTokenAddress == USDC_ADDRESS;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

3. In the event that the function `mint` is executed successfully, the current block timestamp must be within the configured start and end times.

```
/// #if_succeeds {:msg "In the event that the function `mint` is executed
successfully, the current block timestamp must be within the configured
start and end times."} block.timestamp >= old(config.startTime) &&
block.timestamp <= old(config.endTime);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
```

```
    returns(uint256[] memory)
{
```

Passed.

4. Upon the successful execution of the function `mint`, the total number of minted tokens plus the amount to be minted should not exceed the supply limit.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`mint`, the total number of minted tokens plus the amount to be minted
should not exceed the supply limit."} old(minted) + _amounts <=
old(config.supply);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

5. Should the function `mint` be executed with success, the sender's mint amount plus the amount to be minted should not exceed the user limit.

```
/// #if_succeeds {:msg "Should the function `mint` be executed with
success, the sender's mint amount plus the amount to be minted should not
exceed the user limit."} old(mintAmounts[_msgSender()]) + _amounts <=
old(config.userLimit);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

6. In the case of the function `mint` executing flawlessly, the sender must have sufficient balance of the pay token to cover the mint price.

```
/// #if_succeeds {:msg "In the case of the function `mint` executing
flawlessly, the sender must have sufficient balance of the pay token to
cover the mint price."} IERC20(_payTokenAddress).balanceOf(_msgSender()) >=
old(config.mintPrice) * _amounts;
```

```
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

7. Upon the successful execution of the function `mint`, the sender must have sufficient allowance for the contract to transfer the pay token.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`mint`, the sender must have sufficient allowance for the contract to
transfer the pay token."} IERC20(_payTokenAddress).allowance(_msgSender(),
address(this)) >= old(config.mintPrice) * _amounts;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

8. Should the function `mint` be executed with success, the mint amount for the sender should be updated correctly.

```
/// #if_succeeds {:msg "Should the function `mint` be executed with
success, the mint amount for the sender should be updated correctly."}
mintAmounts[_msgSender()] == old(mintAmounts[_msgSender()]) + _amounts;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

9. In the event that the function `mint` executes successfully, the total number of minted tokens should be updated correctly.

```
/// #if_succeeds {:msg "In the event that the function `mint` executes
successfully, the total number of minted tokens should be updated
correctly."} minted == old(minted) + _amounts;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Violated. Please refer to H-01.

10. Upon the successful execution of the function `mint`, the next token ID should be incremented by the amount of tokens minted.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`mint`, the next token ID should be incremented by the amount of tokens
minted."} nextTokenId == old(nextTokenId) + _amounts;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

11. Should the function `mint` be executed with success, the correct number of tokens should be minted and returned.

```
/// #if_succeeds {:msg "Should the function `mint` be executed with
success, the correct number of tokens should be minted and returned."}
$result.length == _amounts;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

12. In the case of the function `mint` executing flawlessly, the RareshopSKUMinted event should be emitted with the correct parameters.

```
/// #if_succeeds {:msg "In the case of the function `mint` executing
flawlessly, the RareshopSKUMinted event should be emitted with the correct
parameters."} emitted RareshopSKUMinted(_msgSender(), old(config.mintPrice)
* _amounts, $result);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

13. Upon the successful execution of the function `exercisePrivilege`, the privilege with the specified `_privilegeId` for the given `_tokenId` should be marked as exercised by the address `_to`.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`exercisePrivilege`, the privilege with the specified `_privilegeId` for
the given `_tokenId` should be marked as exercised by the address `_to`."}
privilegeExercisedAddresses[_tokenId][_privilegeId] == _to;
function exercisePrivilege(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata _data
    )
    external
    override
    checkPrivilegeId(_privilegeId)
{
```

Passed.

14. Should the function `exercisePrivilege` execute without fail, and in the instance where the parameter `_to` is not the owner of the token with the identifier `_tokenId`, the transaction must revert.

```
/// #if_succeeds {:msg "Should the function `exercisePrivilege` execute
without fail, and in the instance where the parameter `_to` is not the
owner of the token with the identifier `_tokenId`, the transaction must
revert."} _to == _ownerOf(_tokenId);
```

```
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

15. In the event that the function `exercisePrivilege` executes successfully, and if the privilege with the specified `_privilegeId` for the given `_tokenId` has already been exercised, the transaction must revert.

```
/// #if_succeeds {:msg "In the event that the function `exercisePrivilege`
executes successfully, and if the privilege with the specified
`_privilegeId` for the given `_tokenId` has already been exercised, the
transaction must revert."} old(privilegeExercisedAddresses[_tokenId]
[_privilegeId]) == address(0);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

16. Upon the successful execution of the function `exercisePrivilege`, the address `_to` should be added to the list of addresses that have exercised the privilege with the specified `_privilegeId` for the given `_tokenId`.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`exercisePrivilege`, the address `_to` should be added to the list of
addresses that have exercised the privilege with the specified
`_privilegeId` for the given `_tokenId`."} addressExercisedPrivileges[_to]
[_privilegeId].length == old(addressExercisedPrivileges[_to]
[_privilegeId].length) + 1;
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
```

```
    returns(uint256[] memory)
{
```

Passed.

17. Should the function `exercisePrivilege` be executed flawlessly, and in the scenario where the parameter `_privilegeId` represents a postage privilege, the `post` function must be called with the correct parameters.

```
/// #if_succeeds {:msg "Should the function `exercisePrivilege` be executed
flawlessly, and in the scenario where the parameter `_privilegeId`
represents a postage privilege, the `post` function must be called with the
correct parameters."} privileges[_privilegeId].pType == 1 ==> $result ==
post(_msgSender(), _to, _tokenId, _privilegeId, _data);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

18. In the case where the function `exercisePrivilege` executes without error, and if the parameter `_privilegeId` does not represent a postage privilege, the `post` function must not be called.

```
/// #if_succeeds {:msg "In the case where the function `exercisePrivilege`
executes without error, and if the parameter `_privilegeId` does not
represent a postage privilege, the `post` function must not be called."}
privileges[_privilegeId].pType != 1 ==> $result != post(_msgSender(), _to,
_tokenId, _privilegeId, _data);
function mint(
    address _payTokenAddress,
    uint256 _amounts,
    bytes32[] memory _whiteListProof
    )
    external
    checkWhiteList(_whiteListProof)
    returns(uint256[] memory)
{
```

Passed.

19. Upon successful execution of the function `post`, if the parameter `_privilegeId` is a postage privilege, the `privilegeExercisedPostages` for the given `_tokenId` and `_privilegeId` should be updated to the provided `postage` value.

```
/// #if_succeeds {:msg "Upon successful execution of the function `post`,
if the parameter `_privilegeId` is a postage privilege, the
`privilegeExercisedPostages` for the given `_tokenId` and `_privilegeId`
should be updated to the provided `postage` value."}
privileges[_privilegeId].pType == 1 ==> privilegeExercisedPostages[_tokenId]
[_privilegeId] == postage;
function post(
    address _sender,
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata _data
    )
    internal
{
```

Passed.

20. In the event that the function `post` executes without fail, the `privilegeExercisedAddresses` for the specified `_tokenId` and `_privilegeId` should remain unaltered if the `postage` is zero, or should be set to `_to` if the `postage` is non-zero.

```
/// #if_succeeds {:msg "In the event that the function `post` executes
without fail, the `privilegeExercisedAddresses` for the specified
`_tokenId` and `_privilegeId` should remain unaltered if the `postage` is
zero, or should be set to `_to` if the `postage` is non-zero."}
(old(privilegeExercisedAddresses[_tokenId][_privilegeId]) == address(0) &&
postage == 0) || (postage > 0 && privilegeExercisedAddresses[_tokenId]
[_privilegeId] == _to);
function post(
    address _sender,
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata _data
    )
    internal
{
```

Passed.

21. Should the function `post` be executed with success, and in the instance where the `postage` is greater than zero, the balance of the sender's ERC20 token at the beginning of the transaction must be reduced by the `postage` amount after the transaction.

```
/// #if_succeeds {:msg "Should the function `post` be executed with
success, and in the instance where the `postage` is greater than zero, the
```

```
balance of the sender's ERC20 token at the beginning of the transaction
must be reduced by the `postage` amount after the transaction."} postage >
0 ==> old(IERC20(payTokenAddress).balanceOf(_sender)) -
IERC20(payTokenAddress).balanceOf(_sender) == postage;
function post(
    address _sender,
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata _data
    )
    internal
{
```

Passed.

22. Upon the successful execution of the function `post`, if the `postage` is greater than zero, the allowance of the sender's ERC20 token to this contract at the beginning of the transaction must be reduced by the `postage` amount after the transaction.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
`post`, if the `postage` is greater than zero, the allowance of the
sender's ERC20 token to this contract at the beginning of the transaction
must be reduced by the `postage` amount after the transaction."} postage >
0 ==> old(IERC20(payTokenAddress).allowance(_sender, address(this))) -
IERC20(payTokenAddress).allowance(_sender, address(this)) == postage;
function post(
    address _sender,
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata _data
    )
    internal
{
```

Passed.