Sep 24, 2024

ZAN

Security Assessment ImKeyNFT

—

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

ImKeyNFT 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 ImKeyNFT 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 **5 Informational issues** in commit bc95c06.

The project team has **resolved issues described in I-02, I-03 and I-05** in commit 07dc84c. They acknowledged the issues described in I-01 and I-04 and decided to keep no change.

1.2. Project Summary

Project Name	ImKeyNFT
Platform	Mint
Language	Solidity
Codebase	Audit 1: • https://github.com/Rare-Shop/Contract-ImKeyNFT/tree/bc95c06d0620768fb07e1e0c1ef4a154ef0d3f16 Final Audit: • https://github.com/Rare-Shop/Contract-ImKeyNFT/tree/07dc84c60d6782c8d774131dcb8fe60bb7025143

1.3. Assessment Summary

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

1.4. Assessment Scope

ID	File	File Hash
1	/src/contract/ImKeyNFTContract.sol	977d6fe73754b843b9d5b497580c1091

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		

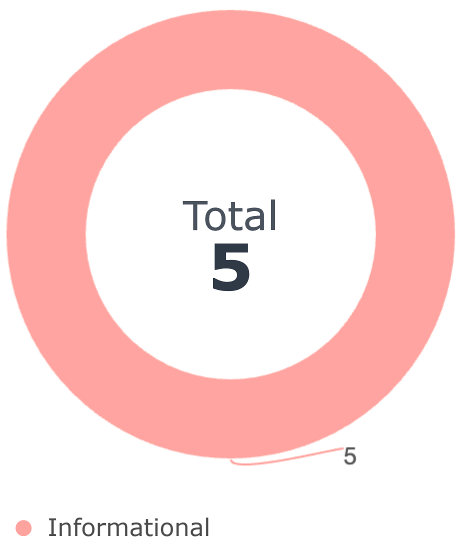
2.2. Optimization Suggestion

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

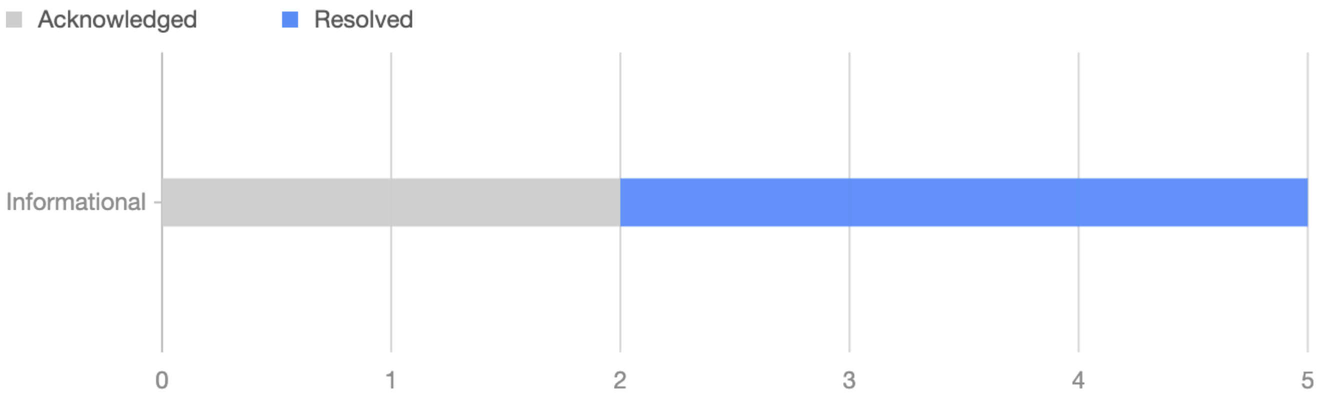
no vulnerabilities.

Optimization Suggestion

Informational: 5

Business Security

no vulnerabilities.



ID	Title	Category	Severity	Status
I-01	Redundant _ownerOf Invocation	Optimization Suggestion	● Informational	Acknowledged
I-02	Optimize Return Value	Optimization Suggestion	● Informational	Resolved
I-03	Function Visibility Can Be External	Optimization Suggestion	● Informational	Resolved
I-04	Set the Constant to Private	Optimization Suggestion	● Informational	Acknowledged
I-05	No Check of Address Params with Zero Address	Optimization Suggestion	● Informational	Resolved

I-01: Redundant _ownerOf Invocation

Informational: Optimization Suggestion

File Location: /src/contract/ImKeyNFTContract.sol:100,102,136,139

Description

The `_requireOwned` function checks that `tokenId` has a present owner and returns said owner. Consequently, invoking the `_ownerOf` function subsequently to obtain the owner of `tokenId` is unnecessary; instead, one should directly utilize the return value of the `_requireOwned` function.

/src/contract/ImKeyNFTContract.sol

```
94      function exercisePrivilege(
95          address _to,
96          uint256 _tokenId,
97          uint256 _privilegeId,
98          bytes calldata
99      ) external override checkPrivilegeId(_privilegeId) {
100         _requireOwned(_tokenId);
101
102         address tokenOwner = _ownerOf(_tokenId);
```

/src/contract/ImKeyNFTContract.sol

```
125     function isExercisable(
126         address _to,
127         uint256 _tokenId,
128         uint256 _privilegeId
129     )
130         external
131         view
132         override
133         checkPrivilegeId(_privilegeId)
134         returns (bool _exercisable)
135     {
136         _requireOwned(_tokenId);
137
138         return
139             _to == _ownerOf(_tokenId) &&
140             tokenPrivilegeAddress[_tokenId] == address(0);
141     }
```

Recommendation

Remove the redundant invocation of the `_ownerOf` function and employ the return value of the `_requireOwned` function instead.

Alleviation

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

I-02: Optimize Return Value

Informational: Optimization Suggestion

File Location: /src/contract/ImKeyNFTContract.sol:163,167

Description

The returned variable `privilegeIds` is specified in the function signature, but it still calls the return statement to return a local variable `output` defined in the function body.

/src/contract/ImKeyNFTContract.sol

```
161  function getPrivilegeIds(
162      uint256 _tokenId
163  ) external view returns (uint256[] memory privilegeIds) {
164      _requireOwned(_tokenId);
165      uint256[] memory output = new uint256[](1);
166      output[0] = PRIVILEGE_ID;
167      return output;
168  }
```

Recommendation

It is recommended to remove the definition of the variable `privilegeIds`.

Alleviation

Resolved in commit 07dc84c. The project team addressed the issue by removing the variable `output` and using the variable `privilegeIds` instead.

I-03: Function Visibility Can Be External

Informational: Optimization Suggestion

File Location: /src/contract/ImKeyNFTContract.sol:170,180

Description

Functions that are not called should be declared as external.

/src/contract/ImKeyNFTContract.sol

```
168     }
169
170     function setMetadataRenderer(address _metadataRenderer) public
        onlyOwner {
171         metadataRenderer = _metadataRenderer;
172     }
```

/src/contract/ImKeyNFTContract.sol

```
178     }
179
180     function setPrivilegeMetadataRenderer(
181         address _privilegeMetadataRenderer
182     ) public onlyOwner {
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

Resolved in commit 07dc84c.

I-04: Set the Constant to Private

Informational: Optimization Suggestion

File Location: /src/contract/ImKeyNFTContract.sol:29,31,33,36,38

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.

/src/contract/ImKeyNFTContract.sol

```
    uint256 private _nextTokenId;

    address public constant PAYMENT_RECEIPIENT_ADDRESS =
        0xC0f068774D46ba26013677b179934Efd7bdefA3F;
    address public constant USDT_ADDRESS =
```

/src/contract/ImKeyNFTContract.sol

```
    address public constant PAYMENT_RECEIPIENT_ADDRESS =
        0xC0f068774D46ba26013677b179934Efd7bdefA3F;
    address public constant USDT_ADDRESS =
        0xED85184DC4BECf731358B2C63DE971856623e056;
    address public constant USDC_ADDRESS =
```

/src/contract/ImKeyNFTContract.sol

```
    address public constant USDT_ADDRESS =
        0xED85184DC4BECf731358B2C63DE971856623e056;
    address public constant USDC_ADDRESS =
        0xBAfC2b82E53555ae74E1972f3F25D8a0Fc4C3682;

```

/src/contract/ImKeyNFTContract.sol

```
        0xBAfC2b82E53555ae74E1972f3F25D8a0Fc4C3682;

    uint256 public constant MINT_PRICE = 60 * 10 ** 6;

    uint256 public constant PRIVILEGE_ID = 1;
```

/src/contract/ImKeyNFTContract.sol

```
    uint256 public constant MINT_PRICE = 60 * 10 ** 6;

    uint256 public constant PRIVILEGE_ID = 1;

    mapping(uint256 tokenId => address to) public tokenPrivilegeAddress;
```

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-05: No Check of Address Params with Zero Address

Informational: Optimization Suggestion

File Location: /src/contract/ImKeyNFTContract.sol:170,180

Description

The input parameter of the address type in the function does not use the zero address for verification.

/src/contract/ImKeyNFTContract.sol

```
168     }
169
170     function setMetadataRenderer(address _metadataRenderer) public onlyOwner {
171         metadataRenderer = _metadataRenderer;
172     }
```

/src/contract/ImKeyNFTContract.sol

```
178     }
179
180     function setPrivilegeMetadataRenderer(
181         address _privilegeMetadataRenderer
182     ) public onlyOwner {
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the address type.

Alleviation

Resolved in commit 07dc84c.

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

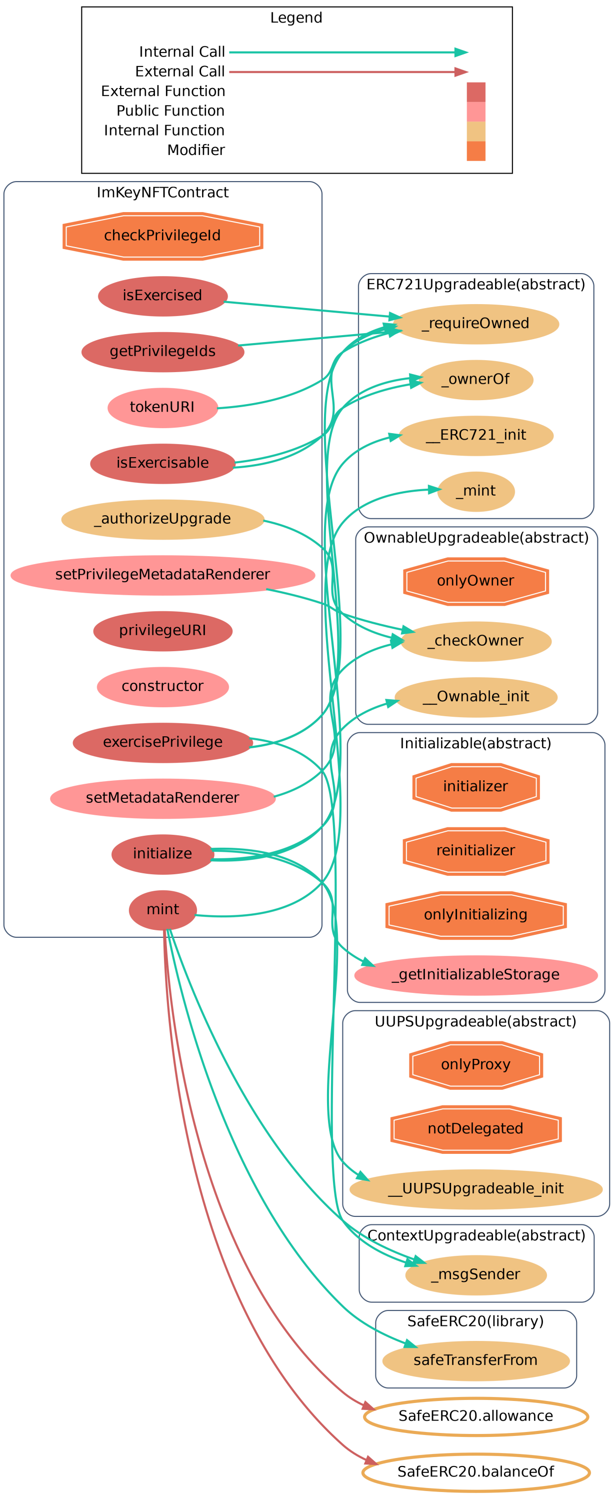
5.1 Visibility

Contract	FuncName	Visibility	Mutability	Modifiers
ImKeyNFTContract	_CTOR_	public	Y	
ImKeyNFTContract	initialize	external	Y	initializer
ImKeyNFTContract	mint	external	Y	
ImKeyNFTContract	exercisePrivilege	external	Y	checkPrivilegeId
ImKeyNFTContract	isExercisable	external	N	checkPrivilegeId
ImKeyNFTContract	isExercised	external	N	checkPrivilegeId
ImKeyNFTContract	getPrivilegeIds	external	N	
ImKeyNFTContract	setMetadataRenderer	public	Y	onlyOwner
ImKeyNFTContract	tokenURI	public	N	
ImKeyNFTContract	setPrivilegeMetadataRenderer	public	Y	onlyOwner
ImKeyNFTContract	privilegeURI	external	N	checkPrivilegeId
ImKeyNFTContract	_authorizeUpgrade	internal	N	onlyOwner

5. Appendix

5.2 Call Graph

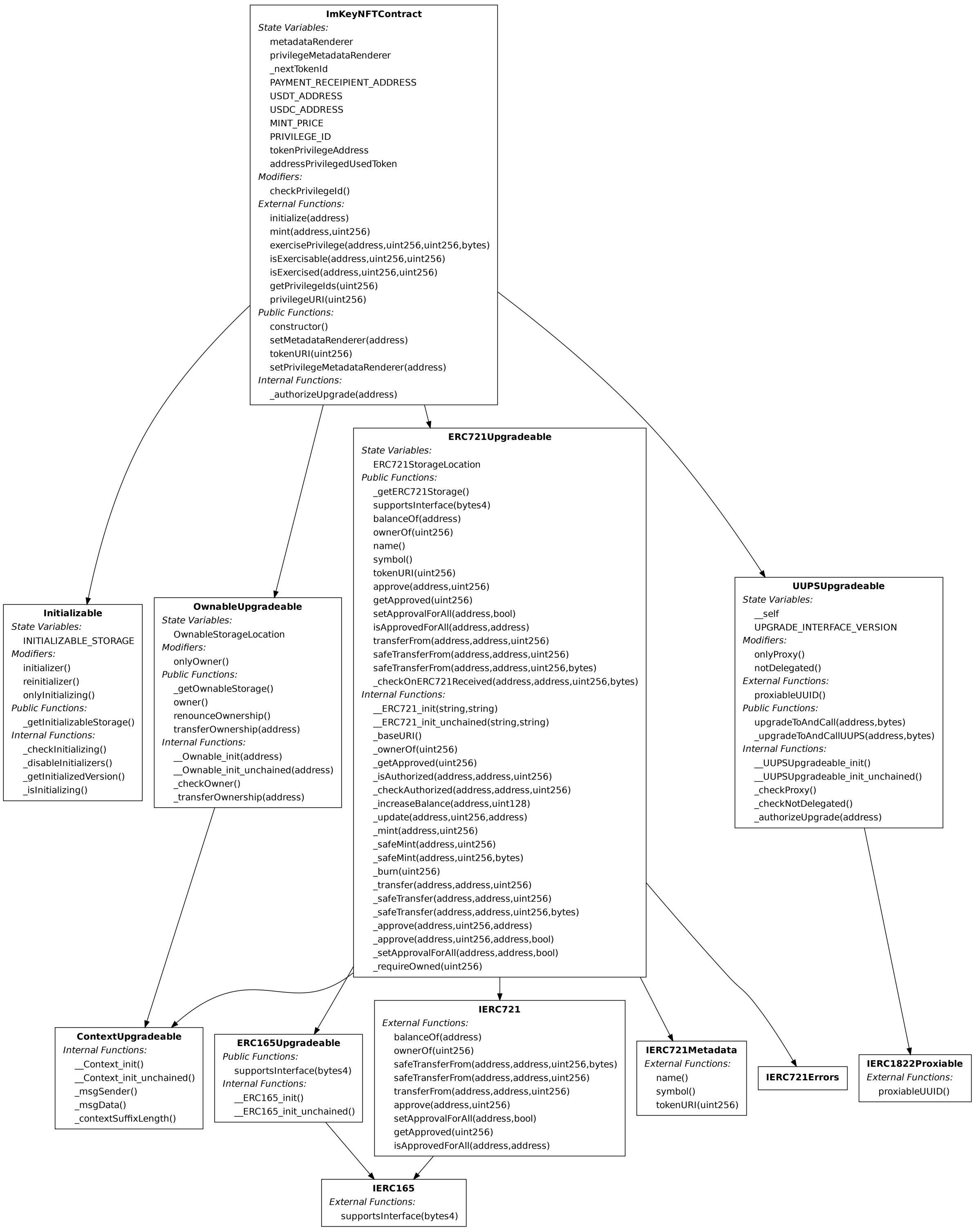
ImKeyNFTContract



5. Appendix

5.3 Inheritance Graph

ImKeyNFTContract



5.4 Formal Verification Metadata

1. The function `mint` should only execute successfully if the caller is attempting to mint between 1 and 10000 tokens, inclusive.

```
/// #if_succeeds {:msg "The function `mint` should only execute
successfully if the caller is attempting to mint between 1 and 10000
tokens, inclusive."} (amounts > 0 && amounts <= 10000) ==> $result;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

2. In the event of a successful mint operation, the transferred payment must correspond to the mint price multiplied by the number of tokens being minted.

```
/// #if_succeeds {:msg "In the event of a successful mint operation, the
transferred payment must correspond to the mint price multiplied by the
number of tokens being minted."} (erc20Token.safeTransferFrom(sender,
PAYMENT_RECEIPIENT_ADDRESS, payPrice) && amounts > 0) ==> payPrice ==
MINT_PRICE * amounts;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

3. The function `mint` is to proceed without error only if the sender has sufficient balance of the payment token to cover the cost of minting.

```
/// #if_succeeds {:msg "The function `mint` is to proceed without error
only if the sender has sufficient balance of the payment token to cover the
cost of minting."} (erc20Token.balanceOf(sender) >= payPrice && amounts >
0) ==> $result;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

4. The mint operation must be triggered with either USDT or USDC as the payment token.

```
/// #if_succeeds {:msg "The mint operation must be triggered with either
USDT or USDC as the payment token."} (payTokenAddress == USDT_ADDRESS ||
payTokenAddress == USDC_ADDRESS) ==> $result;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

5. The allowance for the contract to spend the sender's tokens must be correctly set before minting can proceed.

```
/// #if_succeeds {:msg "The allowance for the contract to spend the
sender's tokens must be correctly set before minting can proceed."}
(erc20Token.allowance(sender, address(this)) >= payPrice) ==> $result;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

6. The next token ID should be incremented by the number of tokens minted.

```
/// #if_succeeds {:msg "The next token ID should be incremented by the
number of tokens minted."} (amounts > 0) ==> _nextTokenId ==
old(_nextTokenId) + amounts;
function mint(address payTokenAddress, uint256 amounts) external {
```

Passed.

7. When exercising privilege, if the function completes successfully, the token's privilege address must update to the provided `_to` address.

```
/// #if_succeeds {:msg "When exercising privilege, if the function
completes successfully, the token's privilege address must update to the
provided `_to` address."} tokenPrivilegeAddress[_tokenId] == _to;
function exercisePrivilege(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata
) external override checkPrivilegeId(_privilegeId) {
```

Passed.

8. Upon successful privilege exercise, the sender must be the same as the token owner.

```
/// #if_succeeds {:msg "Upon successful privilege exercise, the sender must
be the same as the token owner."} _msgSender() == _ownerOf(_tokenId);
function exercisePrivilege(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata
) external override checkPrivilegeId(_privilegeId) {
```

Passed.

9. After exercising privilege, the `_tokenId` must be included in the list of tokens associated with the privileged address `_to`.

```
/// #if_succeeds {:msg "After exercising privilege, the `_tokenId` must be
included in the list of tokens associated with the privileged address
`_to`."} containsElement(addressPrivilegedUsedToken[_to], _tokenId);
function exercisePrivilege(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata
) external override checkPrivilegeId(_privilegeId) {
```

Passed.

10. The function should only proceed if the `_tokenId` has not been previously exercised, ensuring the token privilege address is zero.

```
/// #if_succeeds {:msg "The function should only proceed if the `_tokenId`
has not been previously exercised, ensuring the token privilege address is
zero."} old(tokenPrivilegeAddress[_tokenId]) == address(0);
function exercisePrivilege(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId,
    bytes calldata
) external override checkPrivilegeId(_privilegeId) {
```

Passed.

11. For the exercise privilege check, if the function reports exercisable, `_to` must be the owner of `_tokenId`.

```
/// #if_succeeds {:msg "For the exercise privilege check, if the function
reports exercisable, `_to` must be the owner of `_tokenId`."} _to ==
_ownerOf(_tokenId);
function isExercisable(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId
)
    external
    view
    override
    checkPrivilegeId(_privilegeId)
    returns (bool _exercisable)
{
```

Passed.

12. The token privilege address for `_tokenId` must be zero to be considered exercisable.

```
/// #if_succeeds {:msg "The token privilege address for `_tokenId` must be
zero to be considered exercisable."} tokenPrivilegeAddress[_tokenId] ==
address(0);
function isExercisable(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId
)
    external
    view
    override
    checkPrivilegeId(_privilegeId)
    returns (bool _exercisable)
{
```

Passed.

13. To confirm privilege exercised, `_to` must match the token privilege address of `_tokenId`.

```
/// #if_succeeds {:msg "To confirm privilege exercised, `_to` must match
the token privilege address of `_tokenId`."} _to ==
tokenPrivilegeAddress[_tokenId];
function isExercised(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId
)
    external
    view
    override
    checkPrivilegeId(_privilegeId)
    returns (bool _exercised)
{
```

Passed.

14. The token privilege address for `_tokenId` must not be zero to consider the privilege as exercised.

```
/// #if_succeeds {:msg "The token privilege address for `_tokenId` must not
be zero to consider the privilege as exercised."}
tokenPrivilegeAddress[_tokenId] != address(0);
function isExercised(
    address _to,
    uint256 _tokenId,
    uint256 _privilegeId
)
    external
    view
```

```
        override
        checkPrivilegeId(_privilegeId)
        returns (bool _exercised)
    {
```

Passed.