[Audited]
https://github.com/balajipachai/curiosity/blob/main/erc404/src/Pandora.sol

Scope of Audit

The scope of this audit was to analyze and document the first ERC404 experimental standard token PANDORA smart contracts for quality, security, correctness and any possible vulnerabilities.

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Correctness
- Race conditions / front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide (As it is deprecated)
- Using throw (As it is deprecated)
- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of the smart contracts, care was taken to ensure:

- The overall quality of the code in the contracts.
- Adherence to solidity's best practices.
- Code documentation, comments, mathematical logic, expected behavior etc.
- Efficient usage of gas.
- Code safety from re-entrancy and other well known vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts

Structural Analysis
In this stage, we meticulously analyzed the design patterns and structure of the smart contracts. Our objective was to ensure that the smart contract is meticulously structured to mitigate any potential issues that may arise in the future.

Static Analysis
A comprehensive static analysis of the smart contracts was conducted to identify potential vulnerabilities. This involved leveraging a range of automated tools to thoroughly assess the security posture of the smart contracts.

Code Review / Manual Analysis
Manual code analysis was conducted to uncover new vulnerabilities and validate those identified during static analysis. This involved a meticulous examination of the contracts, ensuring thorough coverage of potential risks. Additionally, the findings from automated analysis were manually verified to ensure accuracy and completeness.

Gas Consumption
We meticulously monitored gas consumption to identify areas for optimization and enhance efficiency. This involved analyzing code execution to pinpoint opportunities for reducing gas usage while maintaining optimal functionality.

Tools and Platforms used for Audit
Foundry, Solidity Visual Developer, Solhint, Slither, Static Analysis, Mythril

Types of Severity

Each issue outlined in this report has been categorized according to its severity level, with four distinct classifications. Below, we have provided detailed explanations for each severity level to ensure clear understanding and appropriate action.

High Severity Issues

A high severity issue or vulnerability indicates a critical risk to your smart contract's security. These issues have the potential to be exploited, posing significant threats to the performance

and functionality of the contract. It is strongly recommended to prioritize the resolution of these issues before deploying the contract in a live environment.

Medium Severity Issues

Medium severity issues typically stem from errors and deficiencies within the smart contract code. While they may not pose immediate critical risks, they have the potential to cause problems and should be addressed to ensure the contract's robustness and reliability.

Low Severity Issues

Low-level severity issues typically have minimal impact and serve as warnings rather than critical vulnerabilities. While they may not require immediate attention, it is advisable to address them at some point in the future to maintain the overall integrity and quality of the smart contract.

Informational

These levels indicate improvement requests, general questions, cosmetic or documentation errors, or requests for information. They have a low-to-no impact on the functionality or security of the smart contract.

Types of Issues

Open

Security vulnerabilities have been identified and must be resolved; however, they remain unresolved at present.

Resolved

These are the issues identified during the initial audit and have been successfully resolved.

Acknowledged

These are vulnerabilities that have been acknowledged but are pending resolution.

Partially Resolved

Significant efforts have been made to mitigate the risk/impact of the security issue, but it has not been fully resolved yet.

Introduction

XDaiForeignBridge inherits from ForeignBridgeErcToNative, SavingsDaiConnector, and GSNForeignERC20Bridge. The contract appears to be part of a bridge mechanism that allows for the transfer of ERC20 tokens (specifically DAI) between different blockchains or layers, with additional functionality for interacting with a savings mechanism (sDAI vault).

Issues Found

Medium Severity Issues

1. Incorrect erc721 interface

   Incorrect return values for ERC721 functions. A contract compiled with solidity > 0.4.22 interacting with these functions will fail to execute them, as the return value is missing.

Pandora has incorrect ERC721 function interface:ERC404.approve(address,uint256)

```
 /// @notice Function for token approvals
 /// @dev This function assumes id / native if amount less than or equal to current max id
    function approve(
        address spender,
        uint256 amountOrId
    ) public virtual returns (bool) {
        if (amountOrId <= minted && amountOrId > 0) {
            //ERC721-Approval
            address tokenOwner = _ownerOf[amountOrId];

            if (
                msg.sender != tokenOwner &&
                !isApprovedForAll[tokenOwner][msg.sender]
            ) {
                revert Unauthorized();
            }

            getApproved[amountOrId] = spender;

            emit Approval(tokenOwner, spender, amountOrId);
        } else {
            // ERC20-Approval
            allowance[msg.sender][spender] = amountOrId;

            emit Approval(msg.sender, spender, amountOrId);
        }
        return true;
    }
```

ERC721's approve

```
/**
  * @dev See {IERC721-approve}.
*/
function approve(address to, uint256 tokenId) public virtual {
```

```
    _approve(to, tokenId, _msgSender());
}
```

`Pandora.approve(address spender, uint256 amountOrId) public virtual returns (bool)` returns a bool unlike ERC721's approve. Bob deploys the token. Alice creates a contract that interacts with it but assumes a correct ERC721 interface implementation. Alice's contract is unable to interact with Bob's contract.

Recommendation:

Set the appropriate return values and value types for the defined ERC721 functions.

2. Tautology

Pandora.tokenURI(uint256) contains a tautology or contradiction:

- seed <= 255

```
function tokenURI(uint256 id) public view override returns (string memory)
{
        if (bytes(baseTokenURI).length > 0) {
            return string.concat(baseTokenURI, Strings.toString(id));
        } else {
            uint8 seed = uint8(bytes1(keccak256(abi.encodePacked(id))));
            string memory image;
            string memory color;

            if (seed <= 100) {
                image = "1.gif";
                color = "Green";
            } else if (seed <= 160) {
                image = "2.gif";
                color = "Blue";
            } else if (seed <= 210) {
                image = "3.gif";
                color = "Purple";
            } else if (seed <= 240) {
                image = "4.gif";
                color = "Orange";
            } else if (seed <= 255) {
                image = "5.gif";
                color = "Red";
            }
```

```
            string memory jsonPreImage = string.concat(
                string.concat(
                    string.concat('{"name": "Pandora #',
Strings.toString(id)),
                    '","description":"A collection of 10,000 Replicants
enabled by ERC404, an experimental token
standard.","external_url":"https://pandora.build","image":"'
                ),
                string.concat(dataURI, image)
            );
            string memory jsonPostImage = string.concat(
                '","attributes":[{"trait_type":"Color","value":"',
                color
            );
            string memory jsonPostTraits = '"}]}';
            return
                string.concat(
                    "data:application/json;utf8,",
                    string.concat(
                        string.concat(jsonPreImage, jsonPostImage),
                        jsonPostTraits
                    )
                );
        }
    }
```

As `seed` is `uint8` (seed <= 255) will always be true, which is a tautology.

Recommendation:

Fix the incorrect comparison by changing the value type or the comparison.

3. Uninitialized local variables

Pandora.tokenURI(uint256).color is a local variable never initialized

Pandora.tokenURI(uint256).image is a local variable never initialized

```
...
 {
            uint8 seed = uint8(bytes1(keccak256(abi.encodePacked(id))));
            string memory image;
```

```
            string memory color;
...
```

Recommendation:

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Low Severity Issues

1) Reentrancy Benign
    a) Reentrancy in ERC404.safeTransferFrom(address,address,uint256):

External calls:

- to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,) != ERC721Receiver.onERC721Received.selector

State variables written after the call(s):

- transferFrom(from,to,id)
- transferFrom(from,to,id)
    - _owned[from].pop()
    - _owned[to].push(id)
    - _owned[from][_ownedIndex[amountOrId]] = updatedId
    - _owned[from].pop()
    - _owned[to].push(amountOrId)
- transferFrom(from,to,id)
    - delete _ownedIndex[id]
    - _ownedIndex[id] = _owned[to].length - 1
    - _ownedIndex[updatedId] = _ownedIndex[amountOrId]
    - _ownedIndex[amountOrId] = _owned[to].length - 1
- transferFrom(from,to,id)
    - delete _ownerOf[id]
    - _ownerOf[id] = to
    - _ownerOf[amountOrId] = to
- transferFrom(from,to,id)
    - allowance[from][msg.sender] = allowed - amountOrId
- transferFrom(from,to,id)
    - balanceOf[from] -= amount
    - balanceOf[to] += amount
    - balanceOf[from] -= _getUnit()
    - balanceOf[to] += _getUnit()

- transferFrom(from,to,id)
    - delete getApproved[id]
    - delete getApproved[amountOrId]
- transferFrom(from,to,id)
    - minted ++

b) Reentrancy in ERC404.safeTransferFrom(address,address,uint256,bytes):

External calls:

- to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,data) != ERC721Receiver.onERC721Received.selector

State variables written after the call(s):

- transferFrom(from,to,id)
    - _owned[from].pop()
    - _owned[to].push(id)
    - _owned[from][_ownedIndex[amountOrId]] = updatedId
    - _owned[from].pop()
    - _owned[to].push(amountOrId)
- transferFrom(from,to,id)
    - delete _ownedIndex[id]
    - _ownedIndex[id] = _owned[to].length - 1
    - _ownedIndex[updatedId] = _ownedIndex[amountOrId]
    - _ownedIndex[amountOrId] = _owned[to].length - 1
- transferFrom(from,to,id)
    - delete _ownerOf[id]
    - _ownerOf[id] = to
    - _ownerOf[amountOrId] = to
- transferFrom(from,to,id)
    - allowance[from][msg.sender] = allowed - amountOrId
- transferFrom(from,to,id)
    - balanceOf[from] -= amount
    - balanceOf[to] += amount
    - balanceOf[from] -= _getUnit()
    - balanceOf[to] += _getUnit()
- transferFrom(from,to,id)
    - delete getApproved[id]
    - delete getApproved[amountOrId]
- transferFrom(from,to,id)
    - minted ++

Recommendation:

Apply the check-effects-interactions pattern.

2) Reentrancy Events

- Reentrancy in ERC404.safeTransferFrom(address,address,uint256,bytes):
  External calls:
  - to.code.length != 0 &&
    ERC721Receiver(to).onERC721Received(msg.sender,from,id,data) !=
    ERC721Receiver.onERC721Received.selector

  Event emitted after the call(s):

  - ERC20Transfer(from,to,_getUnit())
    - transferFrom(from,to,id)
  - ERC20Transfer(from,to,amount)
    - transferFrom(from,to,id)
  - Transfer(from,address(0),id)
    - transferFrom(from,to,id)
  - Transfer(address(0),to,id)
    - transferFrom(from,to,id)
  - Transfer(from,to,amountOrId)
    - transferFrom(from,to,id)
- Reentrancy in ERC404.safeTransferFrom(address,address,uint256):
  External calls:
  - to.code.length != 0 &&
    ERC721Receiver(to).onERC721Received(msg.sender,from,id,) !=
    ERC721Receiver.onERC721Received.selector

  Event emitted after the call(s):

  - ERC20Transfer(from,to,_getUnit())
    - transferFrom(from,to,id)
  - ERC20Transfer(from,to,amount)
    - transferFrom(from,to,id)
  - Transfer(from,address(0),id)
    - transferFrom(from,to,id)
  - Transfer(address(0),to,id)
    - transferFrom(from,to,id)
  - Transfer(from,to,amountOrId)
    - transferFrom(from,to,id)

Recommendation:

Apply the check-effects-interactions pattern.

Informational Issues

1. Solidity Compiler Versions
    - Pragma version^0.8.0 allows old versions ERC404.sol
    - Pragma version^0.8.0 allows old versions Pandora.sol

Recommendation:

Deploy with any of the following Solidity versions:

- ***0.8.18***

The recommendations take into account:

- ***Risks related to recent releases***
- ***Risks of complex code generation changes***
- ***Risks of new language features***
- ***Risks of known bugs***

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

2. Naming convention
    - Variable ERC404._ownedIndex is not in mixedCase
    - Variable ERC404._owned is not in mixedCase
    - Parameter Pandora.setNameSymbol(string,string)._name is not in mixedCase
    - Parameter Pandora.setTokenURI(string)._tokenURI is not in mixedCase
    - Parameter Pandora.setDataURI(string)._dataURI is not in mixedCase
    - Parameter Pandora.setNameSymbol(string,string)._symbol is not in mixedCase
    - Parameter Ownable.transferOwnership(address)._owner is not in mixedCase
    - Variable ERC404._ownerOf is not in mixedCase

Recommendation:

Follow the Solidity naming convention.

Automated Tests Slither

```
INFO:Detectors:
PandoraTest.test_TransferFromForFT() (test/Pandora.t.sol#257-283) uses arbitrary from in transferFrom: pandora.transferFrom(owner,alice,500000000000000000) (test/Pandora.t.sol#278)
PandoraTest.test_RevertsTransferFromForInvalidSender() (test/Pandora.t.sol#285-292) uses arbitrary from in transferFrom: pandora.transferFrom(alice,bob,2) (test/Pandora.t.sol#291)
PandoraTest.test_RevertsTransferFromWhenToIsAddressZero() (test/Pandora.t.sol#294-301) uses arbitrary from in transferFrom: pandora.transferFrom(bob,address(0),2) (test/Pandora.t.sol#300)
PandoraTest.test_RevertsTransferFromWhenCallerIsUnauthorized() (test/Pandora.t.sol#303-310) uses arbitrary from in transferFrom: pandora.transferFrom(bob,alice,2) (test/Pandora.t.sol#308)
PandoraTest.test_TransferFromForNFT() (test/Pandora.t.sol#312-328) uses arbitrary from in transferFrom: pandora.transferFrom(bob,alice,2) (test/Pandora.t.sol#320)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
PandoraTest.test_Transfer() (test/Pandora.t.sol#202-225) ignores return value by
pandora.transfer(alice,1000000000000000000) (test/Pandora.t.sol#212)
PandoraTest.test_Transfer() (test/Pandora.t.sol#202-225) ignores return value by
pandora.transfer(bob,1000000000000000000) (test/Pandora.t.sol#219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Pandora (src/Pandora.sol#7-87) has incorrect ERC721 function interface:ERC404.approve(address,uint256)
(src/ERC404/ERC404.sol#188-213)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc721-interface
INFO:Detectors:
Pandora.tokenURI(uint256) (src/Pandora.sol#39-86) contains a tautology or contradiction:
        - seed <= 255 (src/Pandora.sol#59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
Pandora.tokenURI(uint256).color (src/Pandora.sol#45) is a local variable never initialized
Pandora.tokenURI(uint256).image (src/Pandora.sol#44) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PandoraTest.setUp() (test/Pandora.t.sol#52-58) ignores return value by
pandora.approve(address(this),type()(uint256).max) (test/Pandora.t.sol#56)
PandoraTest.test_RevertOwnerOfForNonExistingToken() (test/Pandora.t.sol#194-200) ignores return value by
pandora.ownerOf(1) (test/Pandora.t.sol#198)
PandoraTest.test_ApproveForNFT() (test/Pandora.t.sol#227-248) ignores return value by pandora.approve(alice,2)
(test/Pandora.t.sol#236)
PandoraTest.test_ApproveForNFT() (test/Pandora.t.sol#227-248) ignores return value by
pandora.approve(address(this),2) (test/Pandora.t.sol#243)
PandoraTest.test_RevertsApproveWhenCallerIsNotAuthorized() (test/Pandora.t.sol#250-255) ignores return value by
pandora.approve(alice,2) (test/Pandora.t.sol#254)
PandoraTest.test_TransferFromForFT() (test/Pandora.t.sol#257-283) ignores return value by
pandora.approve(bob,1000000000000000000) (test/Pandora.t.sol#264)
PandoraTest.testFuzz_TokenURI(uint8) (test/Pandora.t.sol#330-333) ignores return value by
pandora.tokenURI(tokenId) (test/Pandora.t.sol#332)
PandoraTest.test_SafeTransferFromForFT() (test/Pandora.t.sol#335-361) ignores return value by
pandora.approve(bob,1000000000000000000) (test/Pandora.t.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in ERC404.safeTransferFrom(address,address,uint256) (src/ERC404/ERC404.sol#288-301):
        External calls:
        - to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,) !=
ERC721Receiver.onERC721Received.selector (src/ERC404/ERC404.sol#294-296)
        State variables written after the call(s):
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - _owned[from].pop() (src/ERC404/ERC404.sol#392)
                - _owned[to].push(id) (src/ERC404/ERC404.sol#380)
                - _owned[from][_ownedIndex[amountOrId]] = updatedId (src/ERC404/ERC404.sol#257)
                - _owned[from].pop() (src/ERC404/ERC404.sol#259)
                - _owned[to].push(amountOrId) (src/ERC404/ERC404.sol#263)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - delete _ownedIndex[id] (src/ERC404/ERC404.sol#393)
                - _ownedIndex[id] = _owned[to].length - 1 (src/ERC404/ERC404.sol#381)
                - _ownedIndex[updatedId] = _ownedIndex[amountOrId] (src/ERC404/ERC404.sol#261)
                - _ownedIndex[amountOrId] = _owned[to].length - 1 (src/ERC404/ERC404.sol#265)
```

```
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - delete _ownerOf[id] (src/ERC404/ERC404.sol#394)
                - _ownerOf[id] = to (src/ERC404/ERC404.sol#379)
                - _ownerOf[amountOrId] = to (src/ERC404/ERC404.sol#252)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - allowance[from][msg.sender] = allowed - amountOrId (src/ERC404/ERC404.sol#273)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - balanceOf[from] -= amount (src/ERC404/ERC404.sol#331)
                - balanceOf[to] += amount (src/ERC404/ERC404.sol#334)
                - balanceOf[from] -= _getUnit() (src/ERC404/ERC404.sol#246)
                - balanceOf[to] += _getUnit() (src/ERC404/ERC404.sol#249)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - delete getApproved[id] (src/ERC404/ERC404.sol#395)
                - delete getApproved[amountOrId] (src/ERC404/ERC404.sol#253)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
                - minted ++ (src/ERC404/ERC404.sol#370)
Reentrancy in ERC404.safeTransferFrom(address,address,uint256,bytes) (src/ERC404/ERC404.sol#304-319):
        External calls:
        - to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,data) !=
ERC721Receiver.onERC721Received.selector (src/ERC404/ERC404.sol#311-313)
        State variables written after the call(s):
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - _owned[from].pop() (src/ERC404/ERC404.sol#392)
                - _owned[to].push(id) (src/ERC404/ERC404.sol#380)
                - _owned[from][_ownedIndex[amountOrId]] = updatedId (src/ERC404/ERC404.sol#257)
                - _owned[from].pop() (src/ERC404/ERC404.sol#259)
                - _owned[to].push(amountOrId) (src/ERC404/ERC404.sol#263)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - delete _ownedIndex[id] (src/ERC404/ERC404.sol#393)
                - _ownedIndex[id] = _owned[to].length - 1 (src/ERC404/ERC404.sol#381)
                - _ownedIndex[updatedId] = _ownedIndex[amountOrId] (src/ERC404/ERC404.sol#261)
                - _ownedIndex[amountOrId] = _owned[to].length - 1 (src/ERC404/ERC404.sol#265)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - delete _ownerOf[id] (src/ERC404/ERC404.sol#394)
                - _ownerOf[id] = to (src/ERC404/ERC404.sol#379)
                - _ownerOf[amountOrId] = to (src/ERC404/ERC404.sol#252)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - allowance[from][msg.sender] = allowed - amountOrId (src/ERC404/ERC404.sol#273)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - balanceOf[from] -= amount (src/ERC404/ERC404.sol#331)
                - balanceOf[to] += amount (src/ERC404/ERC404.sol#334)
                - balanceOf[from] -= _getUnit() (src/ERC404/ERC404.sol#246)
                - balanceOf[to] += _getUnit() (src/ERC404/ERC404.sol#249)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - delete getApproved[id] (src/ERC404/ERC404.sol#395)
                - delete getApproved[amountOrId] (src/ERC404/ERC404.sol#253)
        - transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
                - minted ++ (src/ERC404/ERC404.sol#370)
Reentrancy in PandoraTest.test_RevertDeployWhenOwnerIsZeroAddress() (test/Pandora.t.sol#69-75):
        External calls:
        - vm.expectRevert(invalidOwnerSelector) (test/Pandora.t.sol#73)
        State variables written after the call(s):
        - pandora = new Pandora(address(0)) (test/Pandora.t.sol#74)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in ERC404.safeTransferFrom(address,address,uint256) (src/ERC404/ERC404.sol#288-301):
	External calls:
	- to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,) !=
ERC721Receiver.onERC721Received.selector (src/ERC404/ERC404.sol#294-296)
	Event emitted after the call(s):
	- ERC20Transfer(from,to,_getUnit()) (src/ERC404/ERC404.sol#268)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
	- ERC20Transfer(from,to,amount) (src/ERC404/ERC404.sol#355)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
	- Transfer(from,address(0),id) (src/ERC404/ERC404.sol#397)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
	- Transfer(address(0),to,id) (src/ERC404/ERC404.sol#383)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
	- Transfer(from,to,amountOrId) (src/ERC404/ERC404.sol#267)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#300)
Reentrancy in ERC404.safeTransferFrom(address,address,uint256,bytes) (src/ERC404/ERC404.sol#304-319):
	External calls:
	- to.code.length != 0 && ERC721Receiver(to).onERC721Received(msg.sender,from,id,data) !=
ERC721Receiver.onERC721Received.selector (src/ERC404/ERC404.sol#311-313)
	Event emitted after the call(s):
	- ERC20Transfer(from,to,_getUnit()) (src/ERC404/ERC404.sol#268)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
	- ERC20Transfer(from,to,amount) (src/ERC404/ERC404.sol#355)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
	- Transfer(from,address(0),id) (src/ERC404/ERC404.sol#397)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
	- Transfer(address(0),to,id) (src/ERC404/ERC404.sol#383)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
	- Transfer(from,to,amountOrId) (src/ERC404/ERC404.sol#267)
		- transferFrom(from,to,id) (src/ERC404/ERC404.sol#318)
Reentrancy in PandoraTest.test_SetWhitelistEmitsWhiteListSet() (test/Pandora.t.sol#186-192):
	External calls:
	- vm.prank(owner) (test/Pandora.t.sol#187)
	- vm.expectEmit() (test/Pandora.t.sol#188)
	Event emitted after the call(s):
	- WhiteListSet(address(2),false) (test/Pandora.t.sol#189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Pragma version^0.8.13 (script/Counter.s.sol#2) allows old versions
Pragma version^0.8.0 (src/ERC404/ERC404.sol#2) allows old versions
Pragma version^0.8.0 (src/Pandora.sol#2) allows old versions
Pragma version^0.8.0 (test/Pandora.t.sol#2) allows old versions
solc-0.8.22 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter Ownable.transferOwnership(address)._owner (src/ERC404/ERC404.sol#25) is not in mixedCase
Variable ERC404._ownerOf (src/ERC404/ERC404.sol#142) is not in mixedCase
Variable ERC404._owned (src/ERC404/ERC404.sol#145) is not in mixedCase
Variable ERC404._ownedIndex (src/ERC404/ERC404.sol#148) is not in mixedCase
Parameter Pandora.setDataURI(string)._dataURI (src/Pandora.sol#20) is not in mixedCase
Parameter Pandora.setTokenURI(string)._tokenURI (src/Pandora.sol#26) is not in mixedCase
Parameter Pandora.setNameSymbol(string,string)._name (src/Pandora.sol#33) is not in mixedCase
Parameter Pandora.setNameSymbol(string,string)._symbol (src/Pandora.sol#34) is not in mixedCase
```

```
Function PandoraTest.test_ConstructorInvocation() (test/Pandora.t.sol#60-67) is not in mixedCase
Function PandoraTest.test_RevertDeployWhenOwnerIsZeroAddress() (test/Pandora.t.sol#69-75) is not in mixedCase
Function PandoraTest.test_SetDataURIWorksAsOwner() (test/Pandora.t.sol#77-88) is not in mixedCase
Function PandoraTest.test_RevertsSetDataURIFailAsNonOwner() (test/Pandora.t.sol#90-97) is not in mixedCase
Function PandoraTest.test_SetTokenURIWorksAsOwner() (test/Pandora.t.sol#99-114) is not in mixedCase
Function PandoraTest.test_RevertsSetTokenURIFailAsNonOwner() (test/Pandora.t.sol#116-123) is not in mixedCase
Function PandoraTest.test_SetNameAndSymbolWorksAsOwner() (test/Pandora.t.sol#125-152) is not in mixedCase
Function PandoraTest.test_SetNameAndSymbolFailAsNonOwner() (test/Pandora.t.sol#154-160) is not in mixedCase
Function PandoraTest.test_SetWhitelistWorksAsOwner() (test/Pandora.t.sol#162-174) is not in mixedCase
Function PandoraTest.test_RevertSetWhitelistWhenTargetIsAddressZero() (test/Pandora.t.sol#176-184) is not in
mixedCase
Function PandoraTest.test_SetWhitelistEmitsWhiteListSet() (test/Pandora.t.sol#186-192) is not in mixedCase
Function PandoraTest.test_RevertOwnerOfForNonExistingToken() (test/Pandora.t.sol#194-200) is not in mixedCase
Function PandoraTest.test_Transfer() (test/Pandora.t.sol#202-225) is not in mixedCase
Function PandoraTest.test_ApproveForNFT() (test/Pandora.t.sol#227-248) is not in mixedCase
Function PandoraTest.test_RevertsApproveWhenCallerIsNotAuthorized() (test/Pandora.t.sol#250-255) is not in
mixedCase
Function PandoraTest.test_TransferFromForFT() (test/Pandora.t.sol#257-283) is not in mixedCase
Function PandoraTest.test_RevertsTransferFromForInvalidSender() (test/Pandora.t.sol#285-292) is not in mixedCase
Function PandoraTest.test_RevertsTransferFromWhenToIsAddressZero() (test/Pandora.t.sol#294-301) is not in
mixedCase
Function PandoraTest.test_RevertsTransferFromWhenCallerIsUnauthorized() (test/Pandora.t.sol#303-310) is not in
mixedCase
Function PandoraTest.test_TransferFromForNFT() (test/Pandora.t.sol#312-328) is not in mixedCase
Function PandoraTest.testFuzz_TokenURI(uint8) (test/Pandora.t.sol#330-333) is not in mixedCase
Function PandoraTest.test_SafeTransferFromForFT() (test/Pandora.t.sol#335-361) is not in mixedCase
Function PandoraTest.test_SafeTransferFromToEOAForNFT() (test/Pandora.t.sol#363-379) is not in mixedCase
Function PandoraTest.test_SafeTransferFromToContractForNFT() (test/Pandora.t.sol#381-399) is not in mixedCase
Function PandoraTest.test_RevertsSafeTransferFromForNFTWhenReceiverIsNotERC721Receiver()
(test/Pandora.t.sol#401-410) is not in mixedCase
Function PandoraTest.test_SafeTransferFromWithDataToEOAForNFT() (test/Pandora.t.sol#412-428) is not in
mixedCase
Function PandoraTest.test_SafeTransferFromWithDataToContractForNFT() (test/Pandora.t.sol#430-448) is not in
mixedCase
Function PandoraTest.test_RevertsSafeTransferFromWithDataForNFTWhenReceiverIsNotERC721Receiver()
(test/Pandora.t.sol#450-459) is not in mixedCase
Function PandoraTest.test_RevertsTransferOwnershipWhenNewOwnerIsAddressZero() (test/Pandora.t.sol#461-466)
is not in mixedCase
Function PandoraTest.test_RevertTransferOwnershipAsNonOwner() (test/Pandora.t.sol#468-472) is not in
mixedCase
Function PandoraTest.test_TransferOwnershipWorksAsOwner() (test/Pandora.t.sol#474-482) is not in mixedCase
Function PandoraTest.test_RevertsRevokeOwnershipAsNonOwner() (test/Pandora.t.sol#484-488) is not in
mixedCase
Function PandoraTest.test_RevokeOwnershipWorksAsOwner() (test/Pandora.t.sol#490-500) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "operator (test/Pandora.t.sol#28)" inNFTReceiver (test/Pandora.t.sol#21-34)
Redundant expression "from (test/Pandora.t.sol#29)" inNFTReceiver (test/Pandora.t.sol#21-34)
Redundant expression "id (test/Pandora.t.sol#30)" inNFTReceiver (test/Pandora.t.sol#21-34)
Redundant expression "data (test/Pandora.t.sol#31)" inNFTReceiver (test/Pandora.t.sol#21-34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
PandoraTest.alice (test/Pandora.t.sol#49) should be constant
PandoraTest.bob (test/Pandora.t.sol#50) should be constant
```

```
PandoraTest.owner (test/Pandora.t.sol#48) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
**THIS CHECKLIST IS NOT COMPLETE**. Use `--show-ignored-findings` to show all the results.
Summary
 - [arbitrary-send-erc20](#arbitrary-send-erc20) (5 results) (High)
 - [unchecked-transfer](#unchecked-transfer) (2 results) (High)
 - [erc721-interface](#erc721-interface) (1 results) (Medium)
 - [tautology](#tautology) (1 results) (Medium)
 - [uninitialized-local](#uninitialized-local) (2 results) (Medium)
 - [unused-return](#unused-return) (8 results) (Medium)
 - [reentrancy-benign](#reentrancy-benign) (3 results) (Low)
 - [reentrancy-events](#reentrancy-events) (3 results) (Low)
 - [solc-version](#solc-version) (5 results) (Informational)
 - [naming-convention](#naming-convention) (41 results) (Informational)
 - [redundant-statements](#redundant-statements) (4 results) (Informational)
 - [constable-states](#constable-states) (3 results) (Optimization)
```

Results

Apart from a few medium severity issues, no major issues were discovered. Some false positive errors were flagged by the tool.

Closing Summary

As ERC404 is an experimental token standard due diligence was made while auditing the contracts and it is observed that the smart contracts are meticulously crafted and do not contain any high severity issues.