SOLIDITY FINANCE

Pixelmon Trainer

Smart Contract Audit Report

AUDIT SUMMARY

Pixelmon Trainer is building a new platform where users can mint NFTs.

For this audit, we reviewed the project team's PixelmonTrainer contract at 0x8A3749936E723325C6b645a0901470cD9E790B94 on the Ethereum Mainnet.

AUDIT FINDINGS

> *No findings were identified.*
>
> *Date: August 30th, 2022.*
>
> *Updated: November 7th, 2022 to reflect the project's Mainnet address.*

CONTRACT OVERVIEW

- *This contract allows users that have been whitelisted by the team to mint NFTs.*
- *The owner can add/remove addresses from the minting whitelist at any time.*
- *Whitelisted users can mint NFTs from first range by specifying a receiver address and a number of NFTs.*
- *The maximum supply for both the first range is 10,000 NFTs.*
- *The owner can specify 20 addresses that will each be minted an NFT from the second range. This functionality can only be performed by the team one time.*
- *There is no cost associated with minting NFTs.*
- *The owner can update the Base URI of the contract at any time.*
- *As the contract is implemented with Solidity v0.8.x, it is protected from overflows/underflows.*
- *The contract complies with the ERC-721 token standard.*

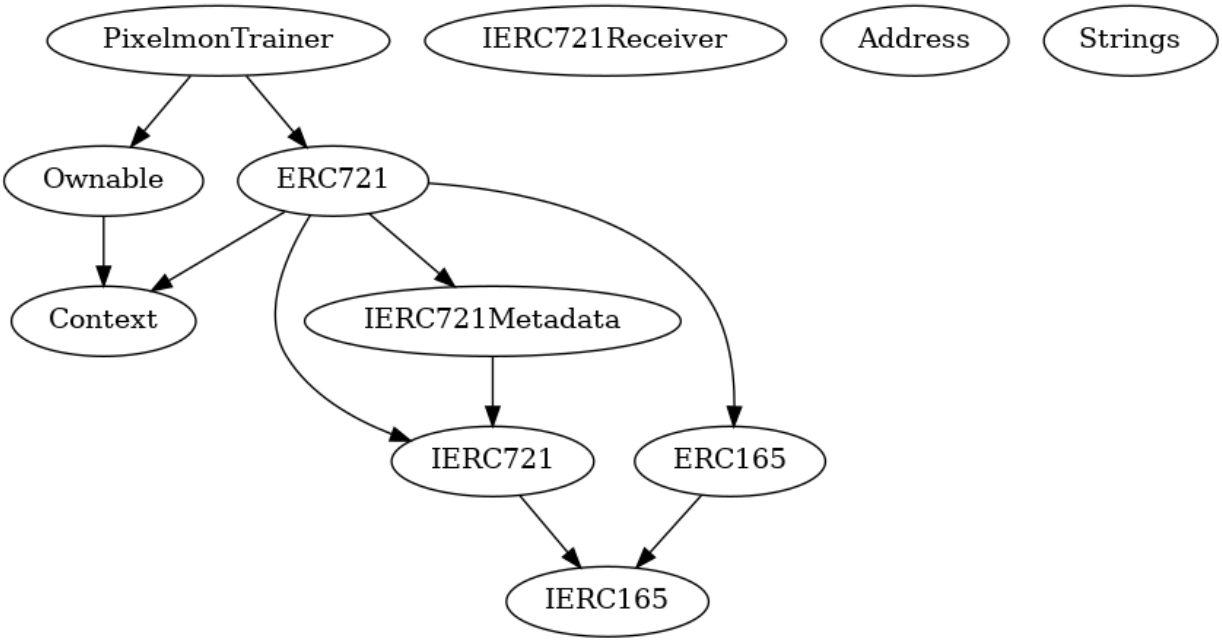
AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary	N/A	PASS

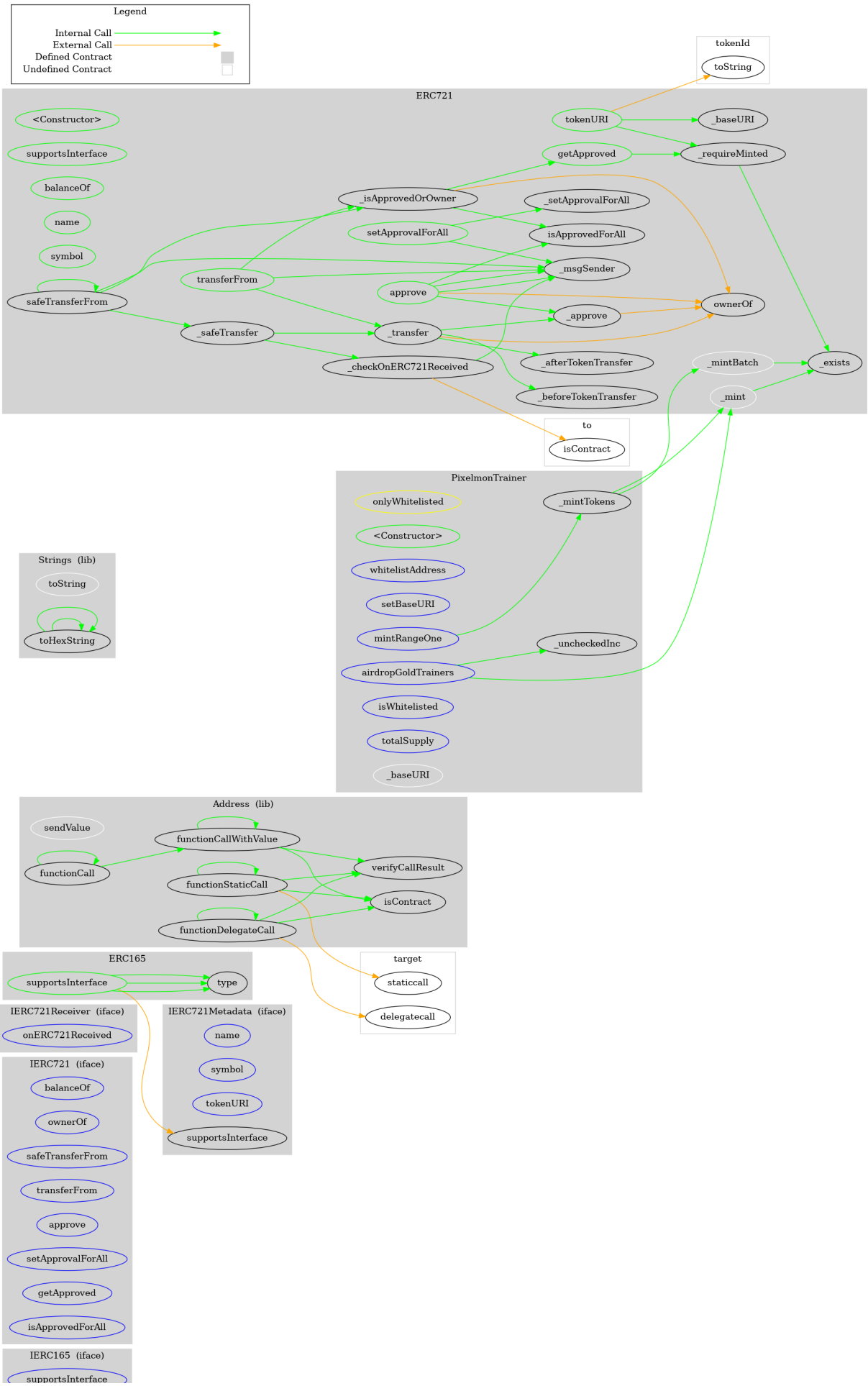
Vulnerability Category	Notes	Result
Write		
Centralization of Control	N/A	PASS
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper	N/A	PASS

Vulnerability Category	Notes	Result
Scheme		
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

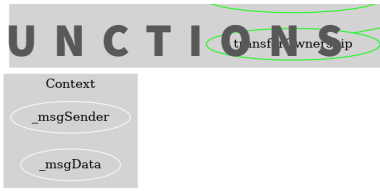
INHERITANCE CHART



FUNCTION GRAPH



FUNCTIONS OVERVIEW




```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  Context
   - [Int] _msgSender
   - [Int] _msgData

+  Ownable (Context)
   - [Pub]  #
   - [Pub] owner
   - [Int] _checkOwner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner
   - [Int] _transferOwnership #

+ [Int] IERC165
   - [Ext] supportsInterface

+ [Int] IERC721 (IERC165)
   - [Ext] balanceOf
   - [Ext] ownerOf
   - [Ext] safeTransferFrom #
```

```
    - [Ext] approve #
    - [Ext] setApprovalForAll #
    - [Ext] getApproved
    - [Ext] isApprovedForAll

+ [Int] IERC721Receiver
    - [Ext] onERC721Received #

+ [Int] IERC721Metadata (IERC721)
    - [Ext] name
    - [Ext] symbol
    - [Ext] tokenURI

+ [Lib] Address
    - [Int] isContract
    - [Int] sendValue #
    - [Int] functionCall #
    - [Int] functionCall #
    - [Int] functionCallWithValue #
    - [Int] functionCallWithValue #
    - [Int] functionStaticCall
    - [Int] functionStaticCall
    - [Int] functionDelegateCall #
    - [Int] functionDelegateCall #
    - [Int] verifyCallResult

+ [Lib] Strings
    - [Int] toString
    - [Int] toHexString
    - [Int] toHexString
    - [Int] toHexString
```

```
+  ERC721 (Context, ERC165, IERC721, IERC721Metadata)
   - [Pub]  #
   - [Pub] supportsInterface
   - [Pub] balanceOf
   - [Pub] ownerOf
   - [Pub] name
   - [Pub] symbol
   - [Pub] tokenURI
   - [Int] _baseURI
   - [Pub] approve #
   - [Pub] getApproved
   - [Pub] setApprovalForAll #
   - [Pub] isApprovedForAll
   - [Pub] transferFrom #
   - [Pub] safeTransferFrom #
   - [Pub] safeTransferFrom #
   - [Int] _safeTransfer #
   - [Int] _exists
   - [Int] _isApprovedOrOwner
   - [Int] _mint #
   - [Int] _mintBatch #
   - [Int] _transfer #
   - [Int] _approve #
   - [Int] _setApprovalForAll #
   - [Int] _requireMinted
   - [Prv] _checkOnERC721Received #
   - [Int] _beforeTokenTransfer #
   - [Int] _afterTokenTransfer #

+  PixelmonTrainer (ERC721, Ownable)
   - [Pub]  #
```

```
        - modifiers: onlyOwner
    - [Ext] setBaseURI #
        - modifiers: onlyOwner
    - [Ext] mintRangeOne #
        - modifiers: onlyWhitelisted
    - [Ext] airdropGoldTrainers #
        - modifiers: onlyOwner
    - [Ext] isWhitelisted
    - [Ext] totalSupply
    - [Prv] _mintTokens #
    - [Int] _baseURI
    - [Int] _uncheckedInc
```

ABOUT SOLIDITY FINANCE

Solidity Finance was founded in 2020 and quickly grew to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1300+ solidity smart contract audits covering all major project types and protocols, securing a total of over $10 billion U.S. dollars in on-chain value.
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.

GO HOME