

Smart Contract

Security Assessment

For Dessaliberty

08 Oct 2024



Ascendant

Ascendant

@ascendantfi

www.ascendant.finance



Ascendant

Table of Contents

3 Disclaimer

4 Executive Summary

5 Overview

6 Findings Summary & Legend

7 Manual Review

- Issue Checking Status
- Audit Findings
- Functional Test Status
- Omitted Results

24 Automated Review

- Unified Model Language

32 Conclusion

DISCLAIMER

This independent audit has been conducted to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by the auditor.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and the auditor is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will the auditor or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Auditor retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Auditor is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. The auditor may, at its discretion, claim bug bounties from third-parties while doing

so.

Executive Summary

The smart contracts reviewed in this audit were found to be **Well Secured**, meaning they contain no critical severity issues that would render them too unsafe to launch. However, it is recommended that the remaining issues found within this report be resolved or mitigated to ensure best user experience.

Security Level	
Well Secured	✓
Secured	
Poorly Secured	
Insecure	

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Hardhat
- Remix IDE
- Slither
- Sol2UML

Overview

This report has been prepared for Dessaliberty for the Ethereum Network. This audit provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.





Summary

Project Name	Dessaliberty
Platform	Ethereum
Language	Solidity





Contracts Assessed

Name	Location
RealEstateStaking.sol	Not Published
RealEstateNFT.sol	Not Published
USDC.sol	Not Published

Findings Summary

Severity	Found
 High	0
 Medium	1
 Low	17
 Informational	56
Total	74

Classification of Issues

 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review



Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Severity	Medium
Contract	RealEstateStaking.sol
Description	Checks-Effects-Interactions: _stake: In 1245
Code Snippet	<pre> /// @dev Staking logic. Override to add custom logic. function _stake(uint256 _tokenId, uint64 _amount) internal virtual { require(_amount != 0, "Staking 0 tokens"); ... isStaking = 2; IERC1155(stakingToken).safeTransfer From(_stakeMsgSender(), address(this), _tokenId, _amount, ""); isStaking = 1; // stakerAddress[_tokenIds[i]] = _stakeMsgSender(); stakers[_tokenId] [_stakeMsgSender()].amountStaked += _amount; </pre>
Recommendation	Function does not follow checks-effects-interactions flow, which requires state updates to come before transfers. This makes the function vulnerable to reentrancy attack, but can be easily mitigated with a Reentrancy Guard.
Status	

Severity	Low
Contract	RealEstateNFT.sol
Description	<p>Checks-Effects-Interactions: mintTo: In 1924</p> <pre> function mintTo(address _to, uint256 _tokenId, string memory _tokenURI, uint256 _amount) public virtual override { uint256 tokenIdToMint; uint256 nextIdToMint = nextTokenIdToMint(); require(_tokenId < nextIdToMint, "invalid id"); tokenIdToMint = _tokenId; require(USDC.transferFrom(msg.send er, address(this), tokenIdPrice[_tokenId]), "Failed to mint with this USDC balance"); _mint(_to, tokenIdToMint, _amount, ""); } </pre>
Code Snippet	
Recommendation	<p>Function does not follow checks-effects-interactions flow due to an external call made during its execution. This makes the function vulnerable to reentrancy attack, but can be easily mitigated with a Reentrancy Guard.</p>
Status	

Severity	Low
Contract	RealEstateNFT.sol
Description	Ignores Return Value withdrawUSDC: In 1933
Code Snippet	<pre>function withdrawUSDC(address to, uint256 amount) public { require(_canMint(), "Not authorized to withdraw."); USDC.transfer(to, amount); }</pre>
Recommendation	The return value of an external transferFrom call is not checked. Ensure that function returns false in case of a revert.
Status	

Severity	Informational
Contract	RealEstateNFT.sol
Description	Token price cannot be changed
Code Snippet	N/A
Recommendation	The token prices cannot be changed after deployment, so no adjustments can be made to the tokenomics. If this is a purposeful feature, disregard this comment.
Status	

Functional Test Status

Function Name	Type/Return Type	Score
RealEstateNFT		
IERC1155		
balanceOf	external	PASS
balanceOfBatch	external	PASS
isApprovedForAll	external	PASS
safeBatchTransferFrom	external	PASS
safeTransferFrom	external	PASS
setApprovalForAll	external	PASS
IERC1155Metadata		
uri	external	PASS
IERC165		
supportsInterface	external	PASS
IERC1155Receiver		
onERC1155BatchReceived	external	PASS
onERC1155Received	external	PASS
Ownable		
_canSetOwner	internal	PASS
_setupOwner	internal	PASS
owner	public	PASS

setOwner	external	PASS
IERC2981		
royaltyInfo	external	PASS
IRoyalty		
getDefaultRoyaltyInfo	external	PASS
getRoyaltyInfoForToken	external	PASS
setDefaultRoyaltyInfo	external	PASS
setRoyaltyInfoForToken	external	PASS
RealEstateNFT		
_canMint	internal	PASS
_canSetContractURI	internal	PASS
burn	external	PASS
burnBatch	external	PASS
constructor	public	PASS
nextTokenIdToMint	public	PASS
_batchMintMetadata	internal	PASS
_freezeBaseURI	internal	PASS
_getBaseURI	internal	PASS
_getBatchId	internal	PASS
_getBatchStartId	internal	PASS
_setBaseURI	internal	PASS
getBaseURICount	public	PASS
getBatchIdAtIndex	public	PASS

_beforeTokenTransfer	internal	PASS
batchMintTo	public	PASS
getUSDCbalance	public	PASS
mintTo	public	PASS
withdrawUSDC	public	PASS
IStaking1155		
claimRewards	external	PASS
getStakeInfo	external	PASS
getStakeInfoForToken	external	PASS
stake	external	PASS
withdraw	external	PASS
Staking1155		
_availableRewards	internal	PASS
_calculateRewards	internal	PASS
_canSetStakeConditions	internal	PASS
_claimRewards	internal	PASS
_mintRewards	internal	PASS
_setDefaultStakingCondition	internal	PASS
_setStakingCondition	internal	PASS
_stake	internal	PASS
_stakeMsgSender	internal	PASS
_updateUnclaimedRewardsForSTaker	internal	PASS
_withdraw	internal	PASS

claimRewards	external	PASS
getDefaultRewardsPerUnitTime	public	PASS
getDefaultTimeUnit	public	PASS
getRewardTokenBalance	external	PASS
getRewardsPerUnitTime	public	PASS
getStakeInfo	external	PASS
getStakeInfoForToken	external	PASS
getTimeUnit	public	PASS
setDefaultRewardsPerUnitTime	external	PASS
setDefaultTimeUnit	external	PASS
setRewardsPerUnitTime	external	PASS
setTimeUnit	external	PASS
stake	external	PASS
withdraw	external	PASS
IWETH		
deposit	external	PASS
transfer	external	PASS
SafeERC20		
_callOptionalReturn	private	PASS
safeApprove	internal	PASS
safeDecreaseAllowance	internal	PASS
safeIncreaseAllowance	internal	PASS
safeTransfer	internal	PASS

safeTransferFrom	internal	PASS
CurrencyTransferLib		
safeTransferERC20	internal	PASS
safeTransferNativeToken	internal	PASS
safeTransferNativeTokenWithWrapper	internal	PASS
transferCurrency	internal	PASS
transferCurrencyWithWrapper	internal	PASS
RealEstateStaking		
_canSetContractURI	internal	PASS
_canSetOwner	internal	PASS
_mintRewards	internal	PASS
_withdrawRewardTokens	internal	PASS
receive	external	PASS
getRewardTokenBalance	external	PASS
depositRewardTokens	external	PASS
withdrawRewardTokens	external	PASS
batchClaimRewards	external	PASS
IERC20		
allowance	external	PASS
approve	external	PASS
balanceOf	external	PASS
totalSupply	external	PASS
transfer	external	PASS

transfeFrom	external	PASS
IERC20Metadata		
decimals	external	PASS
name	external	PASS
symbol	external	PASS
Context		
_msgData	internal	PASS
_msgSender	internal	PASS
ERC20		
_afterTokenTransfer	internal	PASS
_approve	internal	PASS
_beforeTokenTransfer	internal	PASS
_burn	internal	PASS
_mint	internal	PASS
_spendAllowance	internal	PASS
_transfer	internal	PASS
allowance	public	PASS
approve	public	PASS
balanceOf	public	PASS
decreaseAllowance	public	PASS
increaseAllowance	public	PASS
Counters		
current	internal	PASS

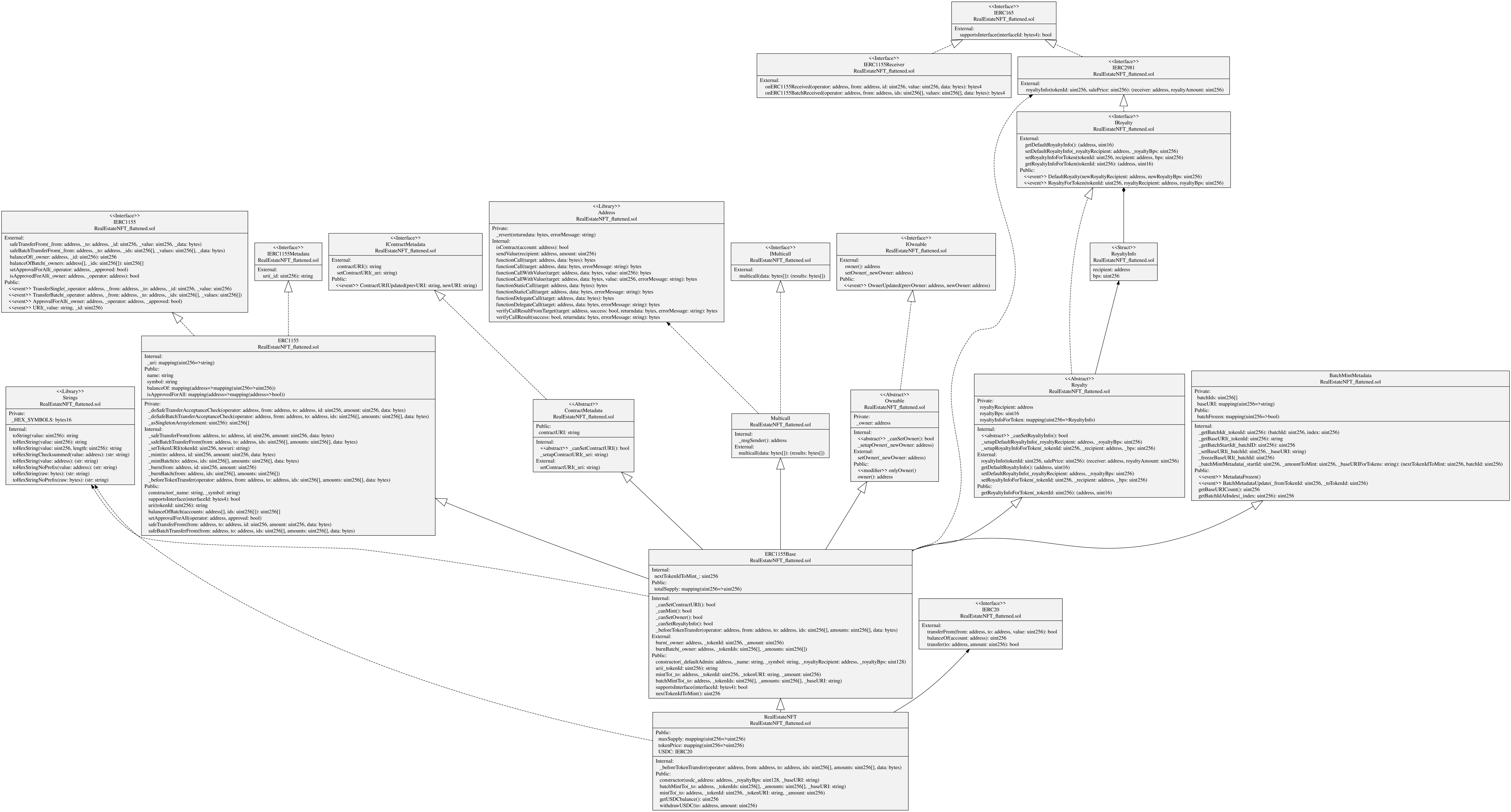
decrement	internal	PASS
increment	internal	PASS
reset	internal	PASS
IMintableERC20		
mintTo	external	PASS
IBurnableERC20		
burn	external	PASS
burnFrom	external	PASS
ERC20Base		
nonce	public	PASS
permit	public	PASS
USDC		
contractURI	external	PASS
constructor	public	PASS

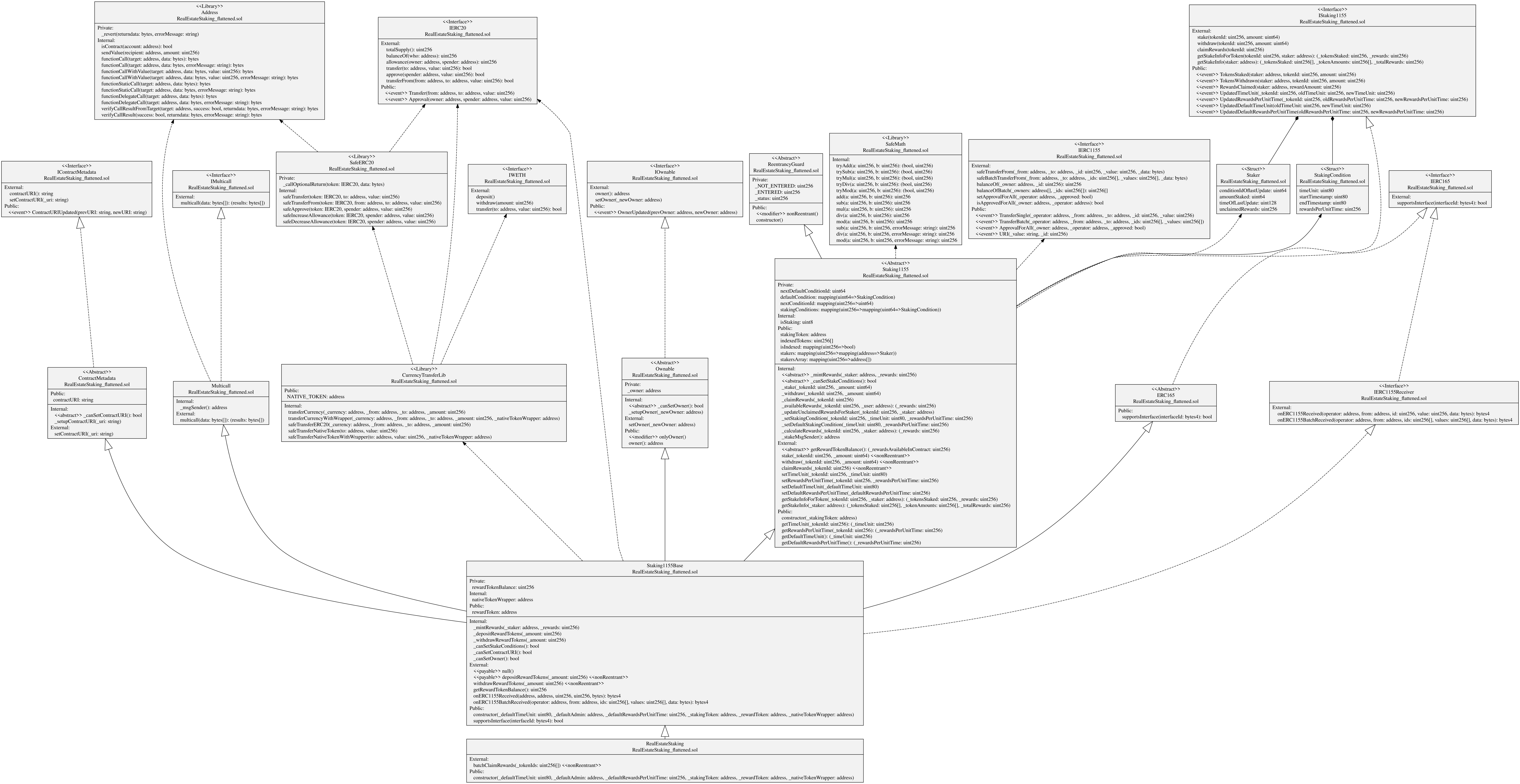
Omitted Results

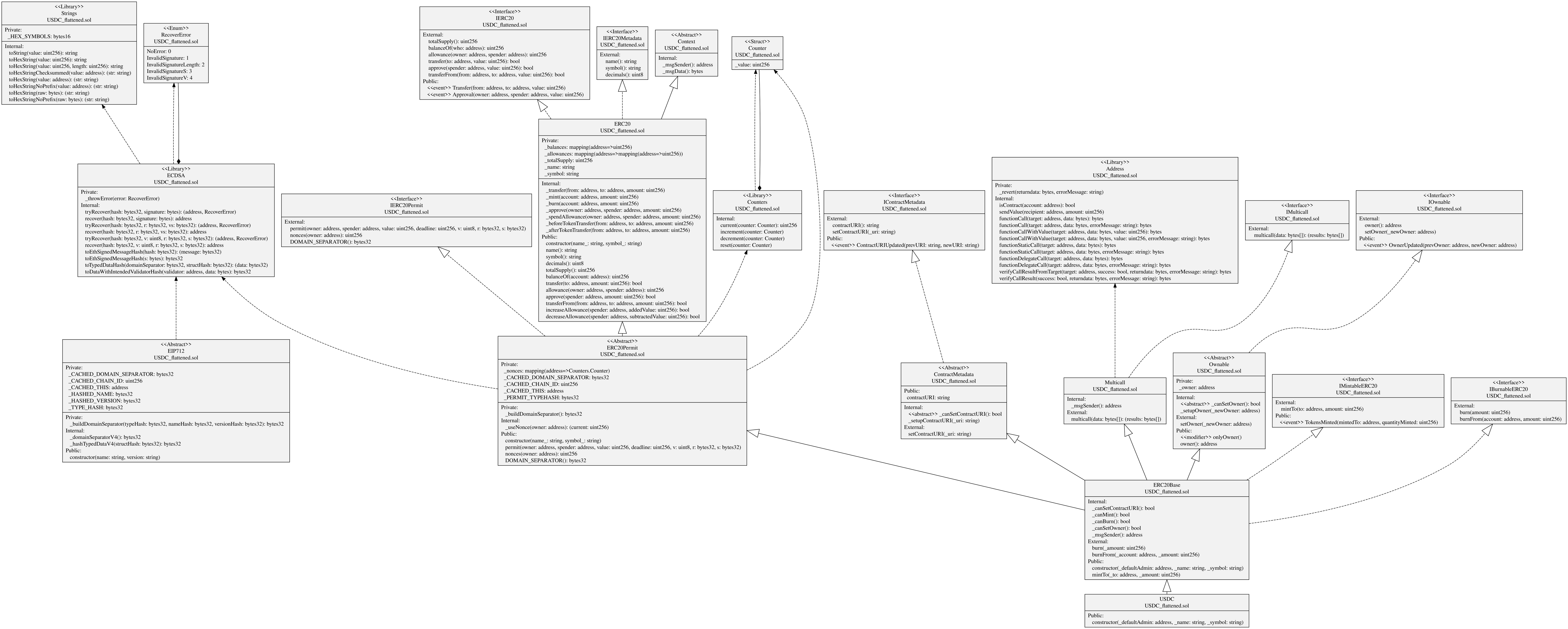
Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.

Automated Review









Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.



Ascendant

Ascendant

@ascendantfi

www.ascendant.finance



Ascendant