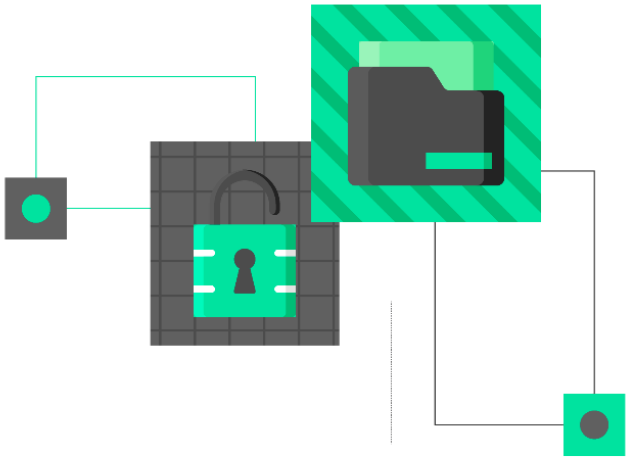
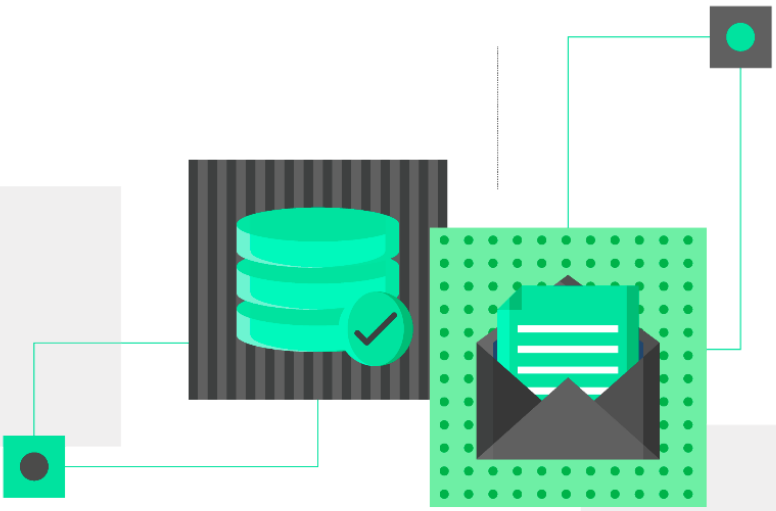
Mantisec Labs

Smart Contract Audit



woowow.io

Dec 2023



Contents

Disclaimer

This disclaimer is to inform you that the report you are reading has been prepared by Mantisec Labs for informational purposes only and should not be considered as investment advice. It is important to conduct your own independent investigation before making any decisions based on the information contained in the report. The report is provided "as is" without any warranties or conditions of any kind and Mantisec Labs excludes all representations, warranties, conditions, and other terms. Additionally, Mantisec Labs assumes no liability or responsibility for any kind of loss or damage that may result from the use of this report.

It is important to note that the analysis in the report is limited to the security of the smart contracts only and no applications or operations were reviewed. The report contains proprietary information, and Mantisec Labs holds the copyright to the text, images, photographs, and other content. If you choose to share or use any part of the report, you must provide a direct link to the original document and credit Mantisec Labs as the author.

By reading this report, you are accepting the terms of this disclaimer. If you do not agree with these terms, it is advisable to discontinue reading the report and delete any copies in your possession.

Audit Process & Methodology

The Mantisec Labs team carried out a thorough audit for the project, starting with an in-depth analysis of code design patterns. This initial step ensured the smart contract's architecture was well-structured and securely integrated with third-party smart contracts and libraries.Also, our team conducted a thorough line-by-line inspection of the smart contract,seeking out potential issues such as Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, among others.
During the Unit testing phase, we assessed the functions authored by the developer to ascertain their precise functionality. Our Automated Testing procedures leveraged proprietary tools designed in-house to spot vulnerabilities and security flaws within the Smart Contract. The code was subjected to an in-depth audit administered by an independent team of auditors, encompassing the following critical aspects:

- Scrutiny of the smart contract's structural analysis to verify its integrity.
- Extensive automated testing of the contract
- A manual line-by-line Code review, undertaken with the aim of evaluating, analyzing, and identifying potential security risks.
- An evaluation of the contract's intended behavior, encompassing a review of provided documentation to ensure the contract conformed to expectations.
- Rigorous verification of storage layout in upgradeable contracts.
- An integral component of the audit procedure involved the identification and recommendation of enhanced gas optimization techniques for the contract

Audit Purpose

Mantisec Labs was hired by the woowow team to review their smart contract.
This audit was conducted in **December 2023**.

The main reasons for this review were:

- To find any possible security issues in the smart contract.
- To carefully check the logic behind the given smart contract.

This report provides valuable information for assessing the level of risk associated with this smart contract and offers suggestions on how to improve its security by addressing any identified issues.

Contract Details

Project Name	woowow
Contract link	https://github.com/X2074/woowow-smart-contracts/tree/main/smart-contracts/
Language	Solidity
Type	ERC20

Security Level Reference

Each problem identified in this report has been categorized into one of the following severity levels:

- **High** severity issues pose significant risks and should be addressed promptly.
- **Medium** severity issues have the potential to create problems and should be on the agenda for future fixes.
- **Low** severity issues are minor concerns and warnings. While they may not require immediate action, addressing them in the future is advisable for overall improvement.

Issues	High	Medium	Low
Open	1	11	36
Closed			

Gas Optimization
15

Findings

Contract Name: AssetShared.sol

G001- Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/AssetShared/AssetShared.sol::111

L002 - Cache Array Length Outside of Loop

Reading array length at each iteration of the loop takes 6 gas (3 for mload and 3 to place memory_offset) in the stack.

Code Location:
~/woowow_contract_audit/contracts/AssetShared/AssetShared.sol::111

G003 - Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than with > 0

Code Location:
~/woowow_contract_audit/contracts/AssetShared/AssetShared.sol::219

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

Contract Name: TokenIdentifiers.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version

Contract Name: CollectionCloneFactory.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

Contract Name: CollectionUpgradeable.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location: ~/woowow_contract_audit/contracts/Collection/CollectionUpgradeable.sol::59

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

Contract Name: DirectListingsLogic.sol

M001 - Missing Input validation on updateListing() function

Impact: High, as it can lead to stuck funds
Likelihood: Low, as it requires user error/misconfiguration

Description:

endTimestamp can be much further in the future than startTimestamp, so duration will be a huge number and the listing may never end.

Recommendations:
Use a minimal duration value, for example 1 day, as well as a max value, for example 20 days.

M002 - The CancelListing() function fails to update the data.totalListing

This will not provide the correct value for totalListings().

M003 - No Filtration for Non-cancelled Listing in GetAllListing

The GetAllListing function lacks filtration for non-cancelled listings, allowing both canceled and active listings to be retrieved without discrimination.

M004 - Refining GetAllValidListings Function: Implementing OnlyExistingListing Modifier for Active Listings

The GetAllValidListings function should only return active listings, and this behavior needs to be ensured by applying the OnlyExistingListing modifier.

M005 - Array Initialization Issue in the ǵetAllValidListings`

The _validListings array is initialized with a length of _listingCount, but the first element of the _validListings array is left empty.

Impact
Users or processes relying on the _validListings array may encounter missing or incomplete data in the returned result set.

```
        _validListings = new Listing[](_listingCount);
        uint256 index = 0;
        uint256 count = _listings.length;
        for (uint256 i = 0; i < count; i += 1) {
            if (_validateExistingListing(_listings[i])) {
                _validListings[index++] = _listings[i];
            }
        }
    }
```

L001 - Cache Array Length Outside of Loop

Reading array length at each iteration of the loop takes 6 gas (3 for mload and 3 to place memory_offset) in the stack.

Code Location:
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::322
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::516

L002 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

L003 - Unnecessary validation for index

Already validated in for-loop cond for (; index < listing.currencies.length; index++) {

L325-L328

```
require(
        index < listing.currencies.length,
        "DirectListing: paying in invalid currency."
    );
```

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::321
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::450
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::452
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::516

G003 - Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than with > 0

Code Location:
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::301
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::506
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::518
~/woowow_contract_audit/contracts/Marketplace/direct-listings/DirectListingsLogic.sol::642

Contract Name: EnglishAuctionLogic.sol

H001 - Missing Authorization check for caller of collectAuctionTokens that must be the winning bidder

Description:

The collectAuctionTokens function lacks an authorization check for the caller, specifically for ensuring that the caller is the winning bidder. This missing authorization check poses a security vulnerability, as it allows unauthorized users to invoke the collectAuctionTokens function, potentially leading to unauthorized access and manipulation of auction tokens. Implementing a proper authorization check is crucial to safeguarding the integrity of the auction system and ensuring that only the designated winning bidder can legitimately collect their tokens.

M001 - Missing Input validation on _validateNewAuction() function can lead to fund loss

Impact: High, as it can lead to stuck funds
Likelihood: Low, as it requires user error/misconfiguration

Description:
There are some problems with the input validation in createAuction function, more specifically related to the timestamp values.

endTimestamp can be much further in the future than startTimestamp, so duration will be a huge number and the auction may never end
timeBufferInSeconds can be much further in the future, so the auction may never end.

Those possibilities should all be mitigated, as they can lead to the initial reserves and/or the bids being stuck in the protocol forever.

Recommendations:
Use a minimal duration value, for example 1 day, as well as a max value, for example 20 days.

M002 - The CancelAuction() function fails to update the data.totalAuctions

This will not provide the correct value for totalAuctions().

M003 - Refining GetAllValidAuctions Function: Implementing OnlyExistingAuction Modifier for Active Auction

The GetAllValidListings function should only return active listings, and this behavior needs to be ensured by applying the OnlyExistingListing modifier.

M004 - Array Initialization Issue in the `getAllValidAuctions`

The _validAuctions array is initialized with a length of _auctionCount, but the first element of the _validAuctions array being left empty.

Impact
Users or processes relying on the _validAuctions array may encounter missing or incomplete data in the returned result set.

```
        _validAuctions = new Auction[](_auctionCount);
        uint256 index = 0;
        uint256 count = _auctions.length;
        for (uint256 i = 0; i < count; i += 1) {
            if (
                _auctions[i].startTimestamp <= block.timestamp &&
                _auctions[i].endTimestamp > block.timestamp &&
                _auctions[i].status == IEnglishAuctions.Status.CREATED &&
                _auctions[i].assetContract != address(0)
            ) {
                _validAuctions[index++] = _auctions[i];
            }
        }
    }
```

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::352
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::354

G003 - Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than with > 0

Code Location:
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::431
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::439
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::442
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::471
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::506
~/woowow_contract_audit/contracts/Marketplace/english-auctions/EnglishAuctionsLogic.sol::658

Contract Name: OffersLogic.sol

M001 - The CancelOffer function requires an update to the data.totalOffers field.

Description

The CancelOffer function is in need of an update that specifically pertains to the data.totalOffers attribute. This modification is essential for ensuring accurate and up-to-date information within the system.

M002 - Array Initialization Issue in the ġetAllValidOfferṡ

The _validOffers array is initialized with a length of _OfferCount, but the first element of the _validOffers array is left empty.

Impact
Users or processes relying on the _validOffers array may encounter missing or incomplete data in the returned result set.

```
        _validOffers = new Offer[](_offerCount);
        uint256 index = 0;
        uint256 count = _offers.length;
        for (uint256 i = 0; i < count; i += 1) {
            if (_validateExistingOffer(_offers[i])) {
                _validOffers[index++] = _offers[i];
            }
        }
    }
```

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:

~/woowow_contract_audit/contracts/Marketplace/offers/OffersLogic.sol::231
~/woowow_contract_audit/contracts/Marketplace/offers/OffersLogic.sol::233

G003 - Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than with > 0

Code Location:
~/woowow_contract_audit/contracts/Marketplace/offers/OffersLogic.sol::275
~/woowow_contract_audit/contracts/Marketplace/offers/OffersLogic.sol::276
~/woowow_contract_audit/contracts/Marketplace/offers/OffersLogic.sol::399

Contract Name: OfferStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ERC2771ContextLogic.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location: ~/woowow_contract_audit/contracts/extension/plugin/ERC2771ContextLogic.sol::15

L002 - Cache Array Length Outside of Loop

Reading array length at each iteration of the loop takes 6 gas (3 for mload and 3 to place memory_offset) in the stack.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/ERC2771ContextLogic.sol::15

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ERC2771ContextStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version

Contract Name: ERC2771ContextConsumer.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ERC2771ContextUpgradeableLogic.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/ERC2771ContextUpgradeableLogic.sol::22

L002 - Cache Array Length Outside of Loop

Reading array length at each iteration of the loop takes 6 gas (3 for mload and 3 to place memory_offset) in the stack.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/ERC2771ContextUpgradeableLogic.sol::22

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ERC2771ContextUpgradeableStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: PermissionEnumerableLogic.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/PermissionsEnumerableLogic.sol::33
~/woowow_contract_audit/contracts/extension/plugin/PermissionsEnumerableLogic.sol::59

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

L005 - Do not use Deprecated Library Functions

The _setupRole method from the AccessControl library is deprecated so it should not be used.

Reference:
https://github.com/OpenZeppelin/openzeppelin-contracts/issues/3918

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/PermissionsEnumerableLogic.sol::79

Contract Name: PermissionsEnumerableStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: PluginMap.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/PluginMap.sol::26
~/woowow_contract_audit/contracts/extension/plugin/PluginMap.sol::48
~/woowow_contract_audit/contracts/extension/plugin/PluginMap.sol::58

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ReentrancyGuardLogic.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: ReentrancyGuardStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: PermissionsLogic.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

L005 - Do not use Deprecated Library Functions

The _setupRole method from the AccessControl library is deprecated so it should not be used.

Reference:
https://github.com/OpenZeppelin/openzeppelin-contracts/issues/3918

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/PermissionsLogic.sol::87
~/woowow_contract_audit/contracts/extension/plugin/PermissionsLogic.sol::128

Contract Name: PermissionsStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: PlatformFeeLogic.sol

L001 - Fee recipient may be address(0)

The recipient of a fee may be address(0), leading to lost ETH.

Code Location:
~/woowow_contract_audit/contracts/extension/plugin/PlatformFeeLogic.sol::60

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: PlatformFeeStorage.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: Router.sol

G001 - Don't Initialize Variables with Default Value

If a variable is not set/initialized, it is assumed to have the default value (0, false, 0x0 etc depending on the data type). If you explicitly initialize it with its default value, you are just wasting gas.

Code Location:

~/woowow_contract_audit/contracts/extension/plugin/Router.sol::166
~/woowow_contract_audit/contracts/extension/plugin/Router.sol::176
~/woowow_contract_audit/contracts/extension/plugin/Router.sol::183
~/woowow_contract_audit/contracts/extension/plugin/Router.sol::206
~/woowow_contract_audit/contracts/extension/plugin/Router.sol::207

~/woowow_contract_audit/contracts/extension/plugin/Router.sol::221
~/woowow_contract_audit/contracts/extension/plugin/Router.sol::228

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: RouterImmutable.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: Marketplace.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.

A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain.

It is recommended to pin to a concrete compiler version.

L005 - Do not use Deprecated Library Functions

The _setupRole method from the AccessControl library is deprecated so it should not be used.

Reference:
https://github.com/OpenZeppelin/openzeppelin-contracts/issues/3918

Code Location:

~/woowow_contract_audit/contracts/Marketplace/entrypoint/Marketplace.sol::63
~/woowow_contract_audit/contracts/Marketplace/entrypoint/Marketplace.sol::64
~/woowow_contract_audit/contracts/Marketplace/entrypoint/Marketplace.sol::65

Contract Name: NFTMetadataRendererLib.sol

G003 - Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than with > 0

Code Location:

~/woowow_contract_audit/contracts/lib/NFTMetadataRendererLib.sol::77
~/woowow_contract_audit/contracts/lib/NFTMetadataRendererLib.sol::78

Contract Name: Create2Deployer.sol

L003 - Unspecific Compiler Version Pragma

Description

Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Contract Name: InitStorage.sol

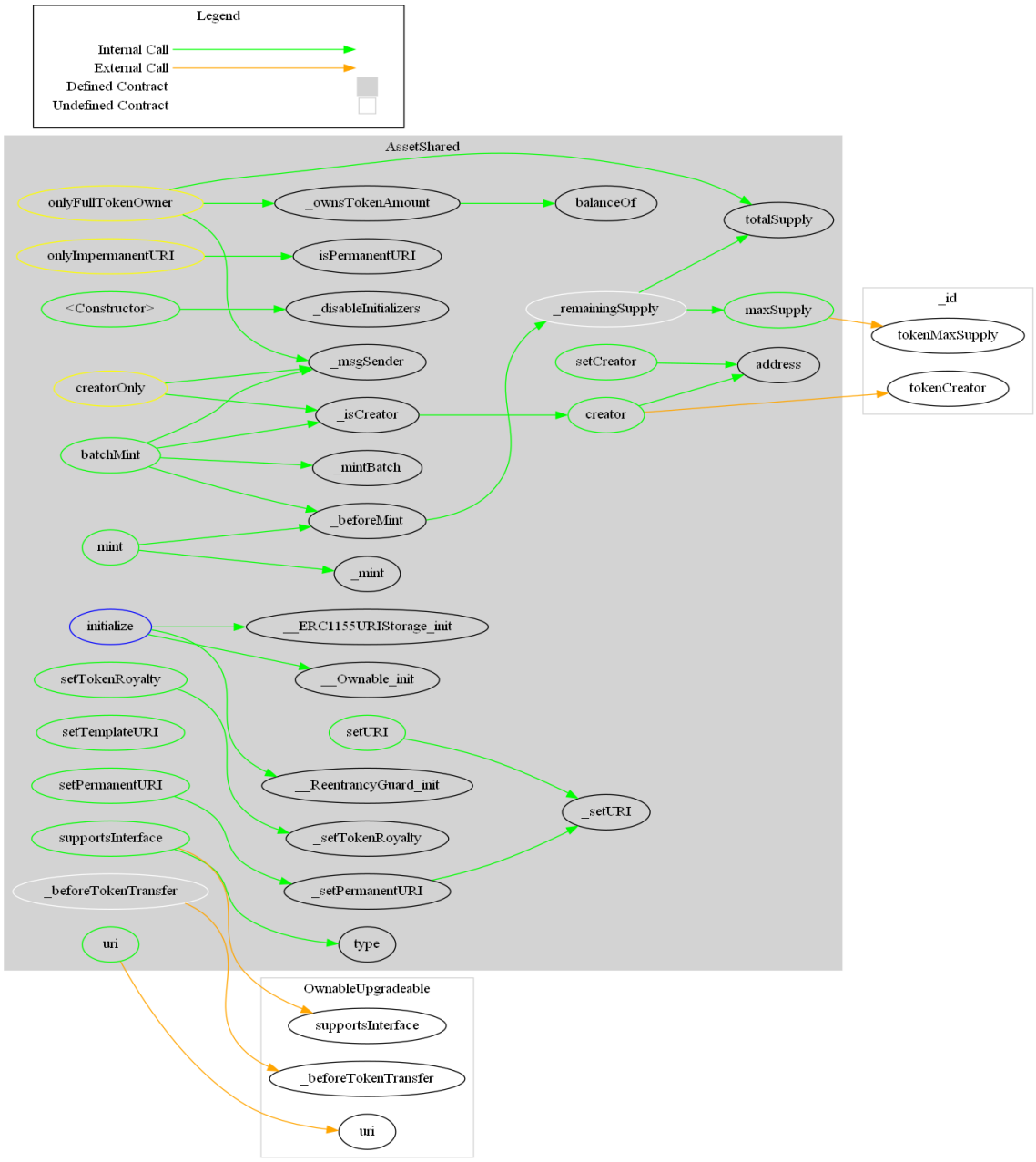
L003 - Unspecific Compiler Version Pragma

Description

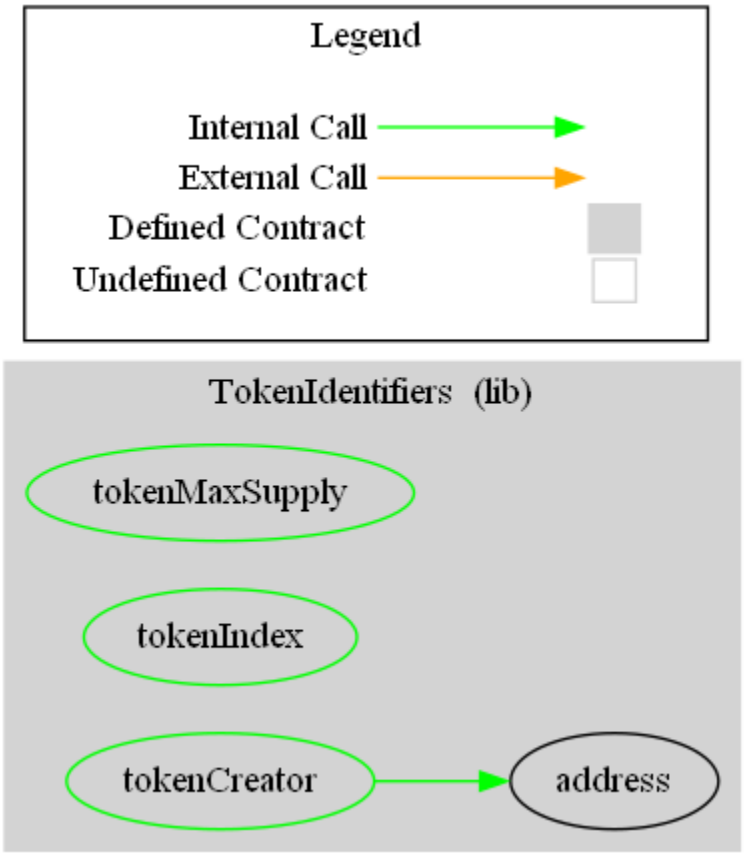
Avoid floating pragmas for non-library contracts. While floating pragmas make sense for libraries to allow them to be included with multiple different versions of applications, it may be a security risk for application implementations.
A known vulnerable compiler version may accidentally be selected or security tools might fall-back to an older compiler version ending up checking a different EVM compilation that is ultimately deployed on the blockchain. It is recommended to pin to a concrete compiler version.

Additional Details

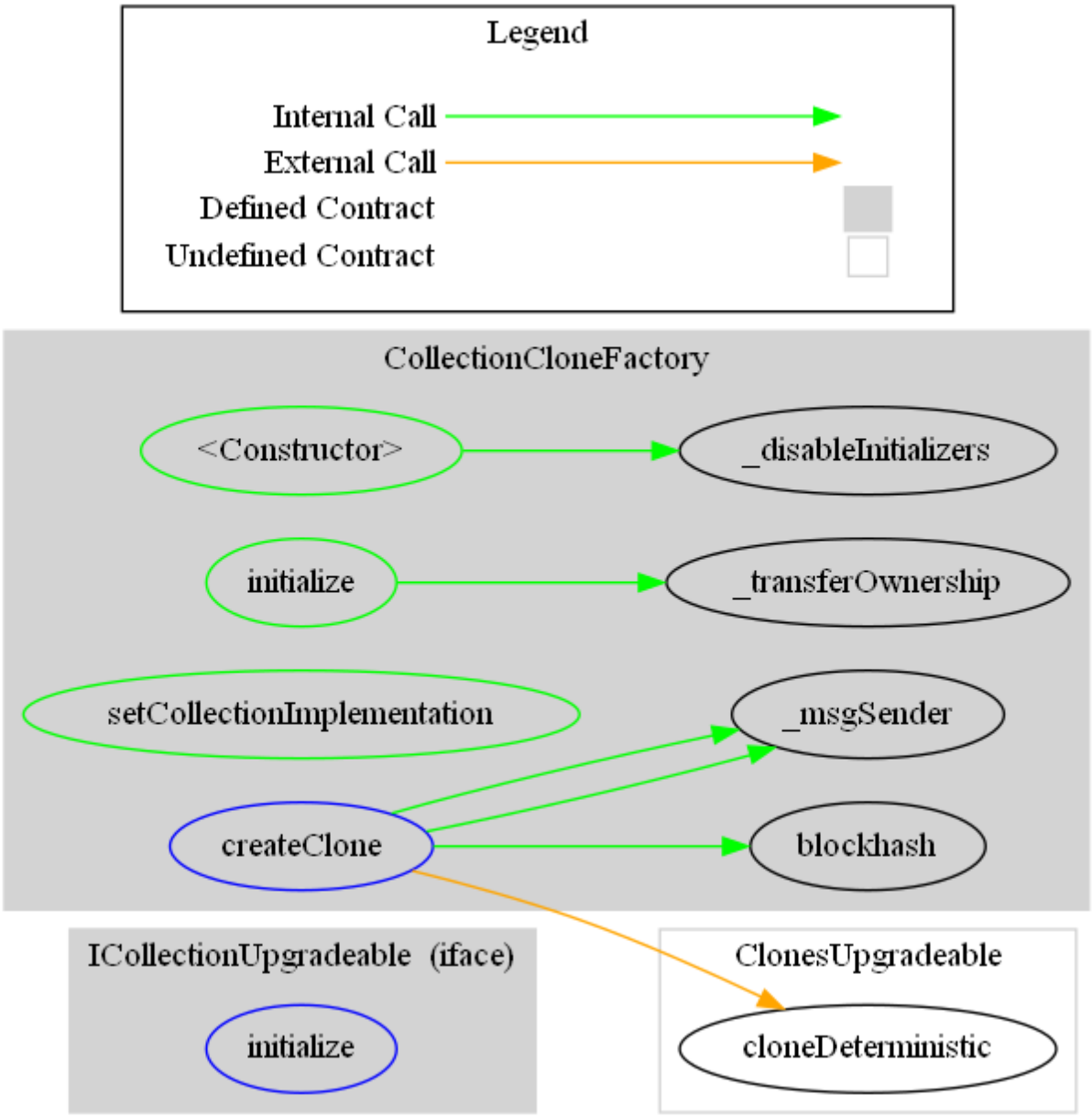
AssetShared



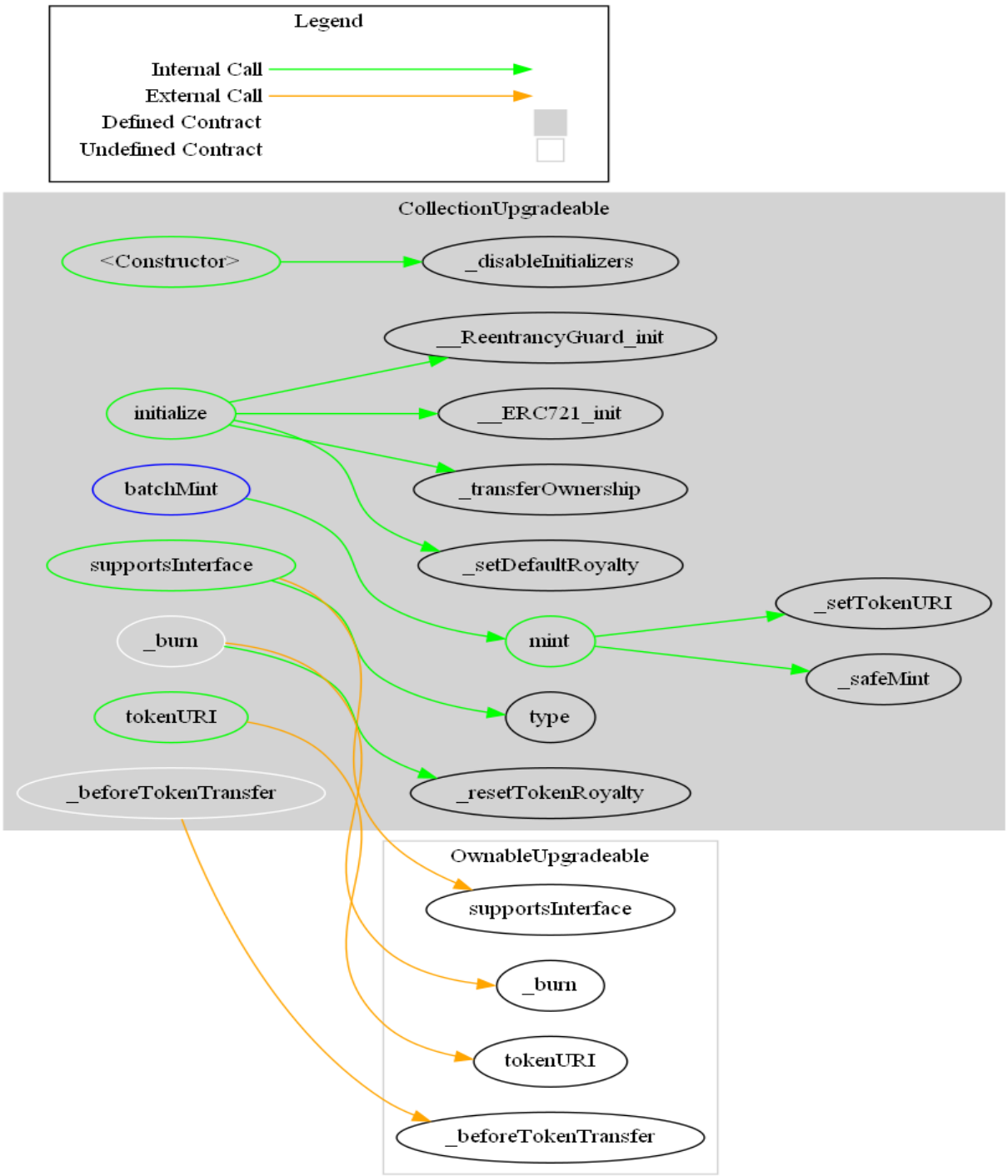
TokenIdentifiers



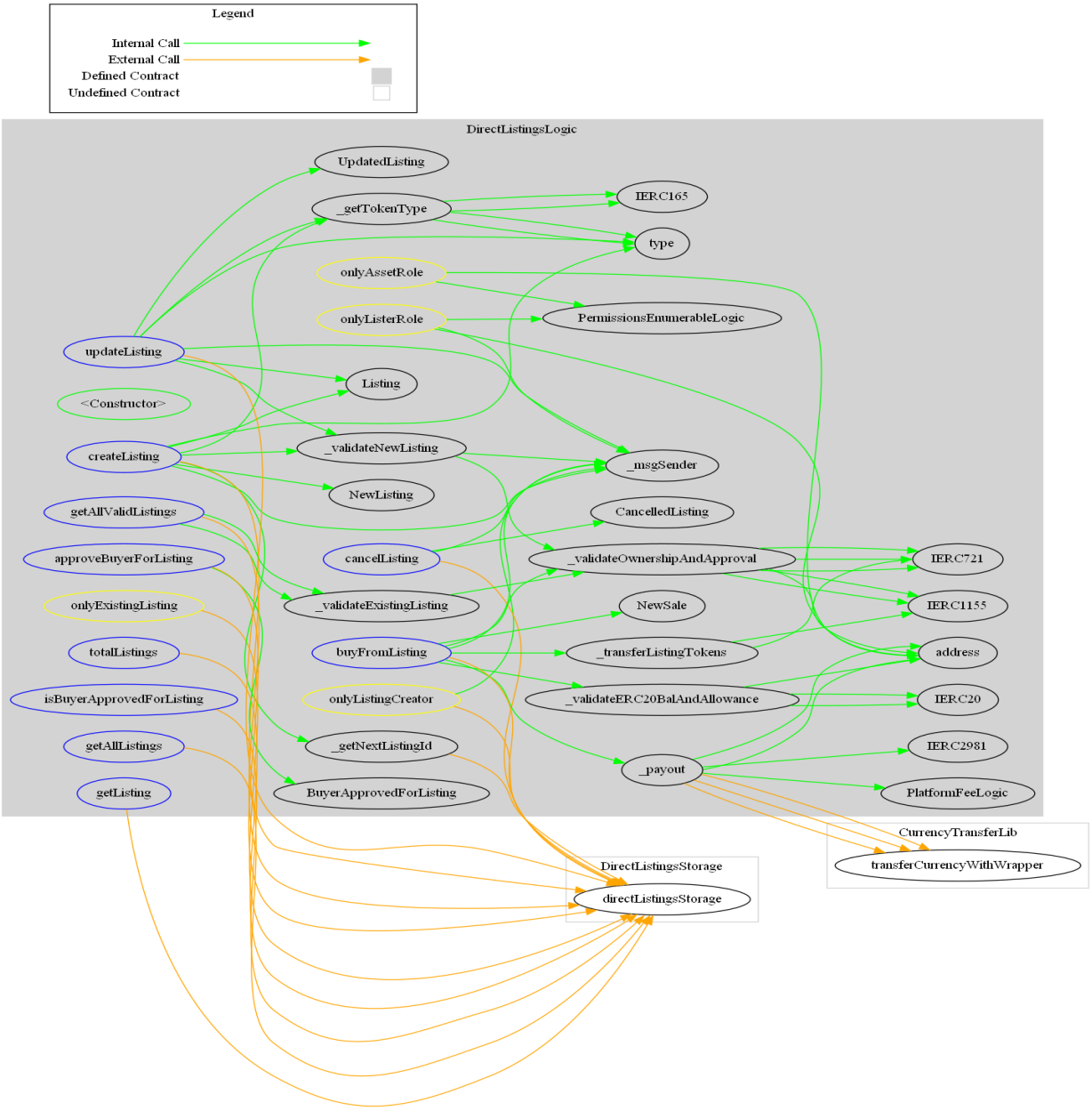
CollectionCloneFactory



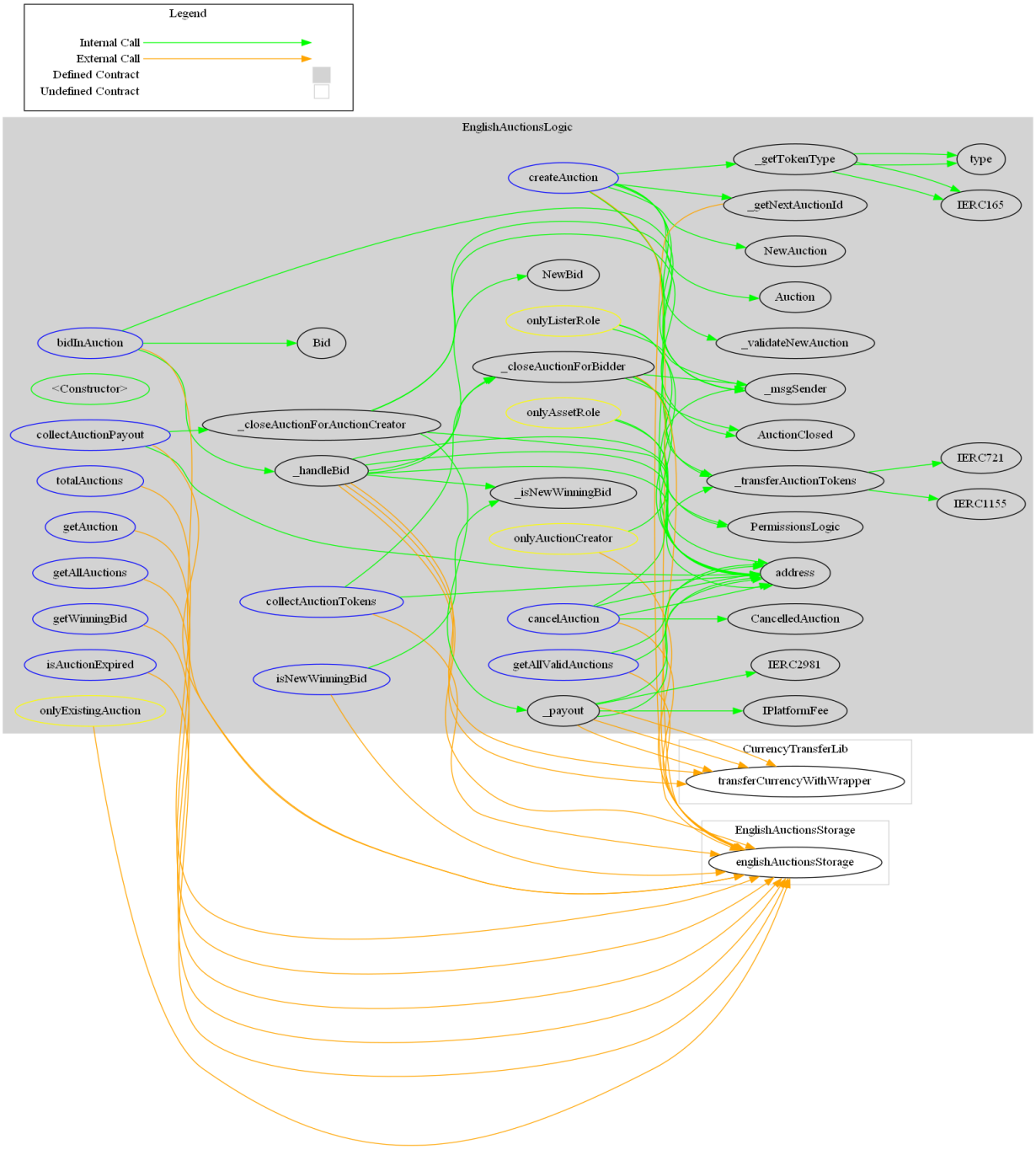
CollectionUpgradeable



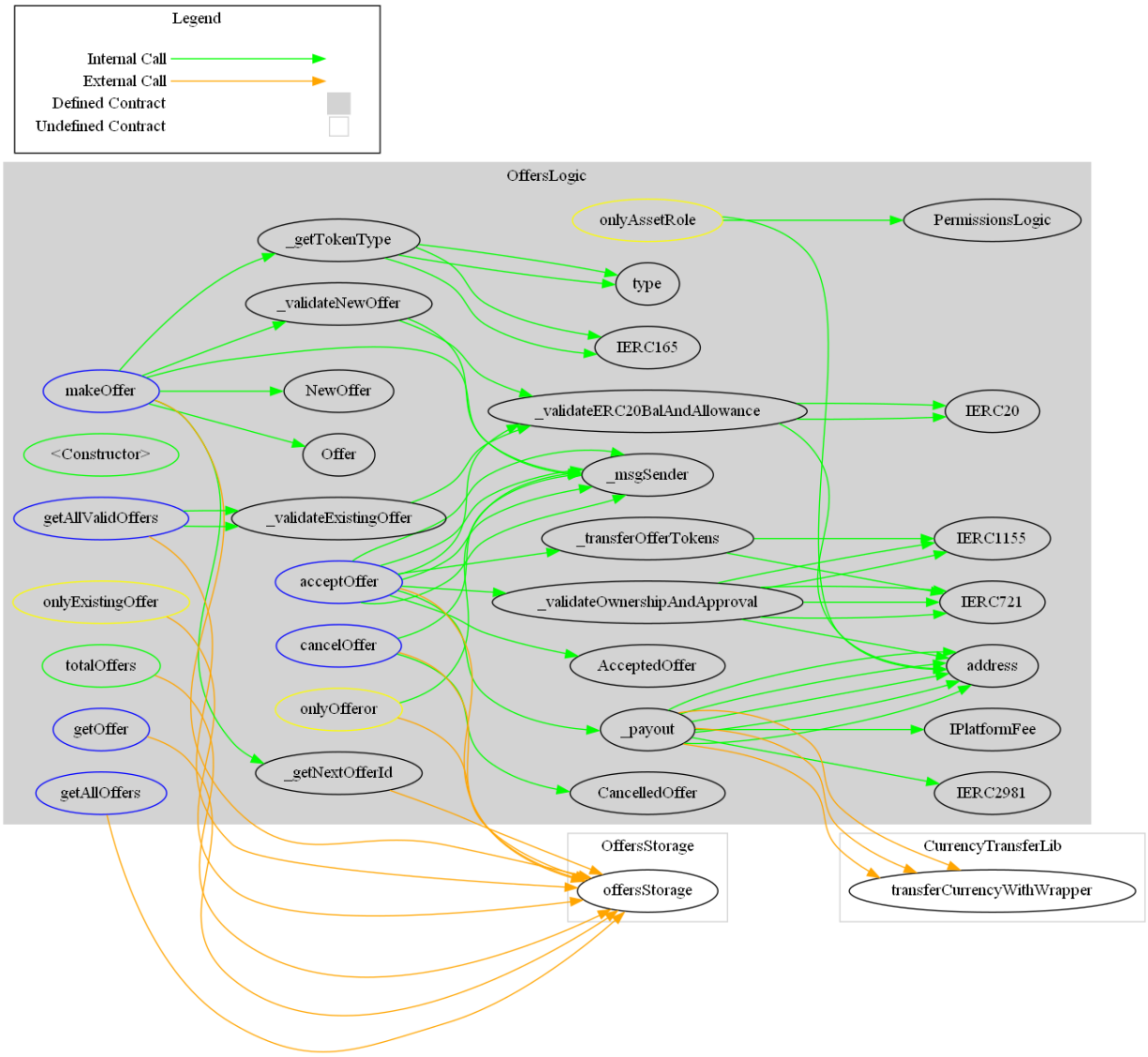
DirectListingsLogic



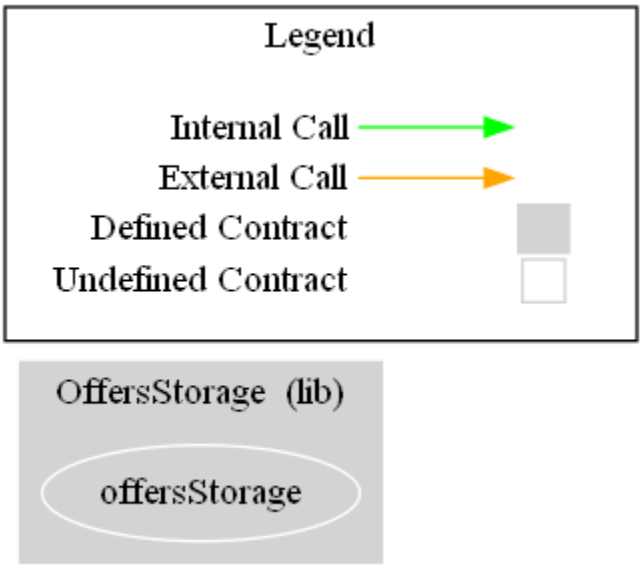
EnglishAuctionLogic



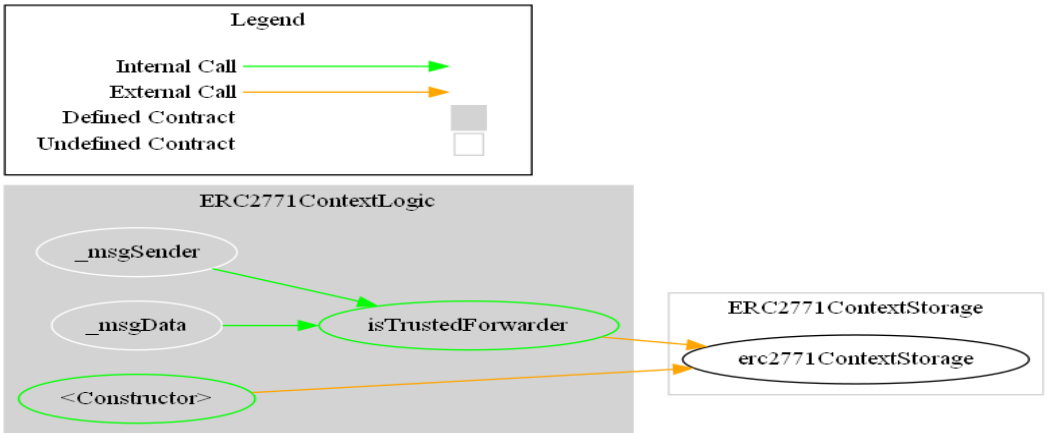
OffersLogic



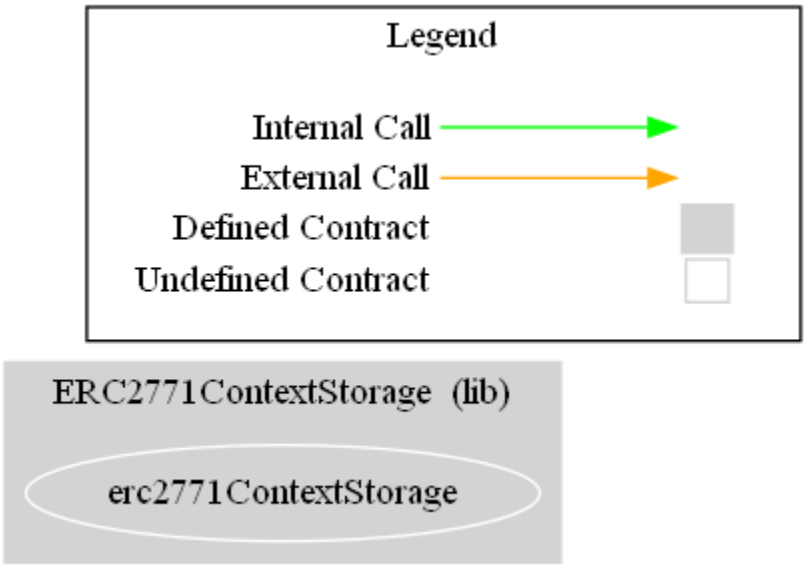
OfferStorage



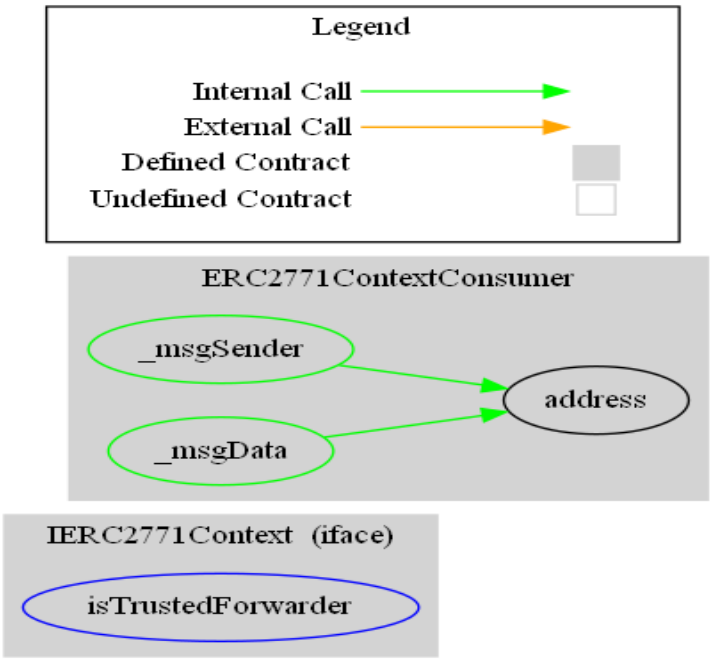
ERC2771ContextLogic



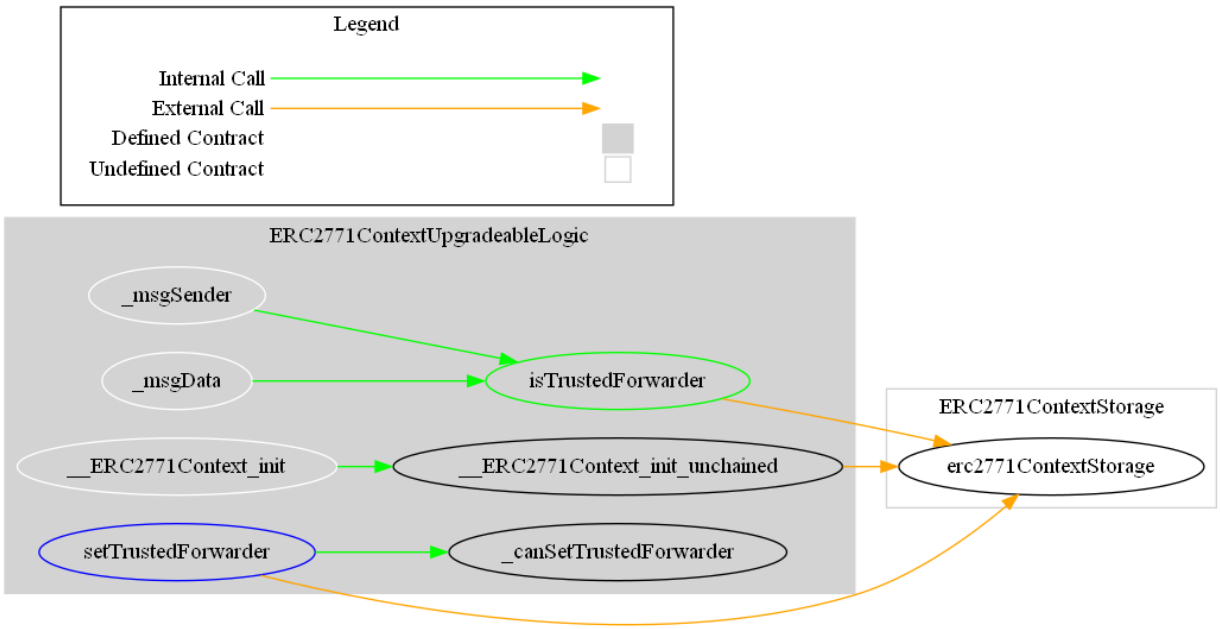
ERC2771ContextStorage



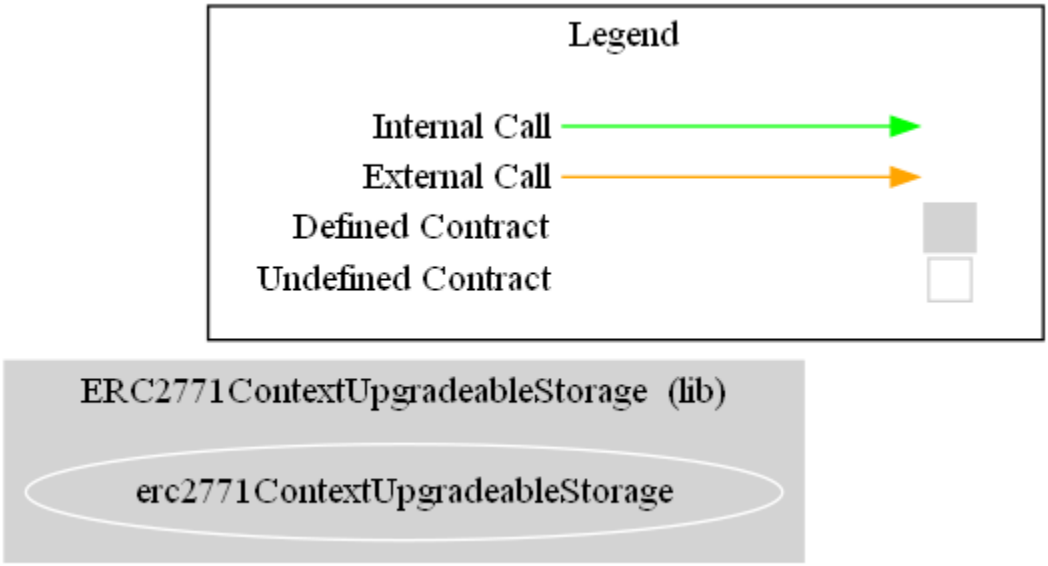
ERC2771ContextConsumer



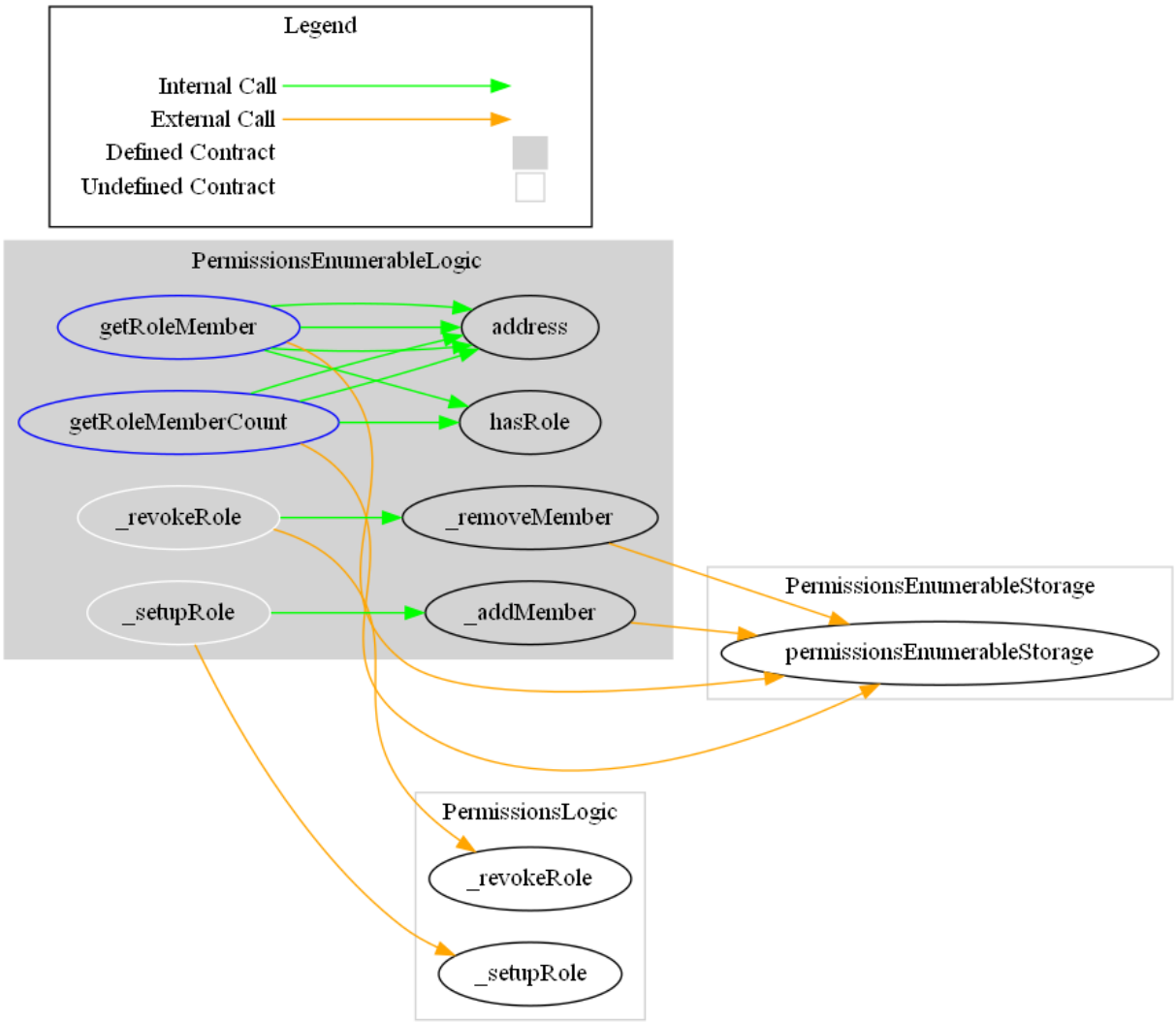
ERC2771ContextUpgradeableLogic



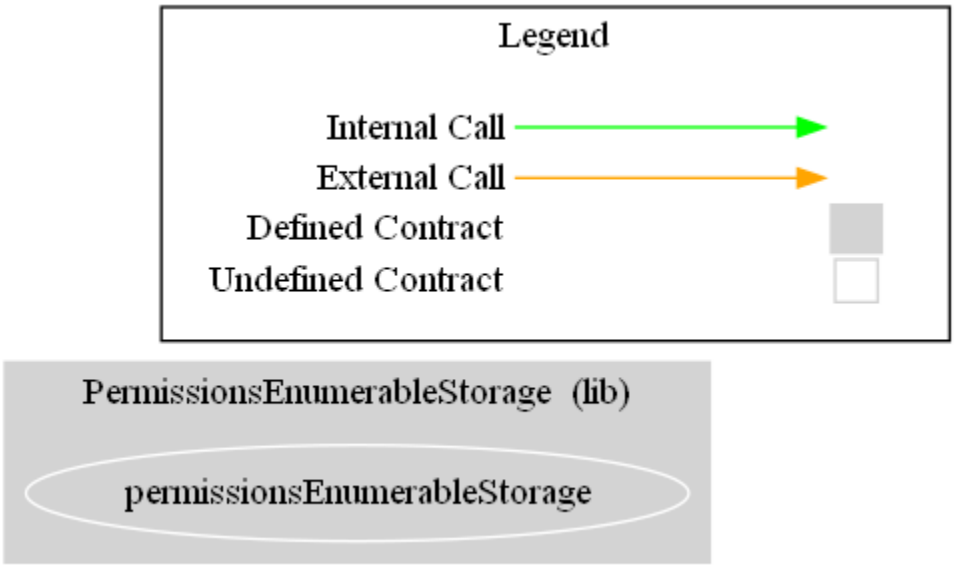
ERC2771ContextUpgradeableStorage



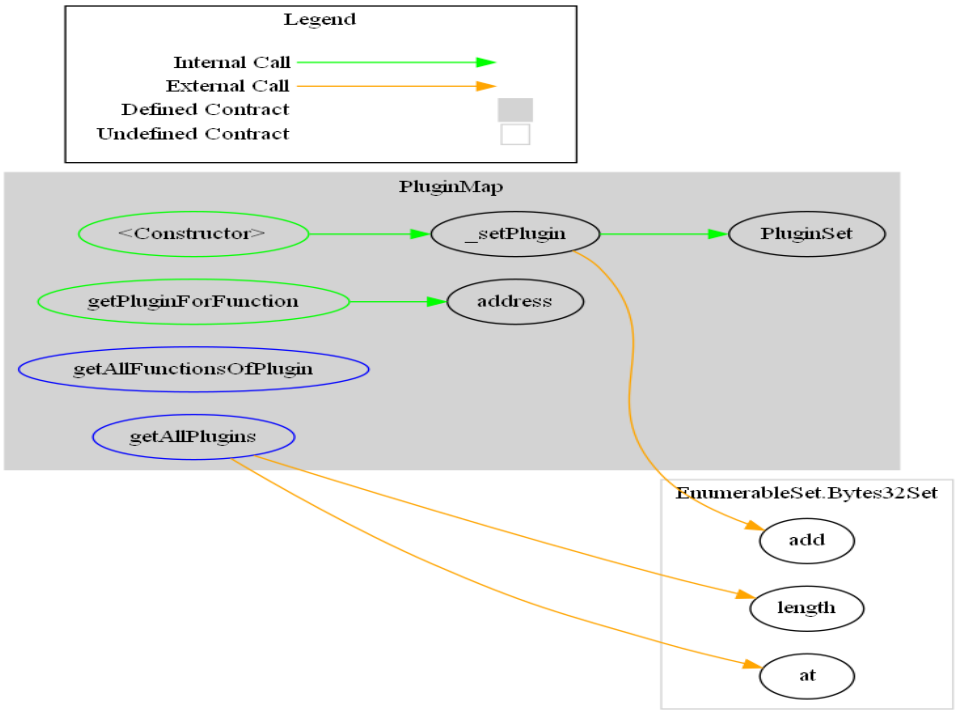
PermissionEnumerableLogic



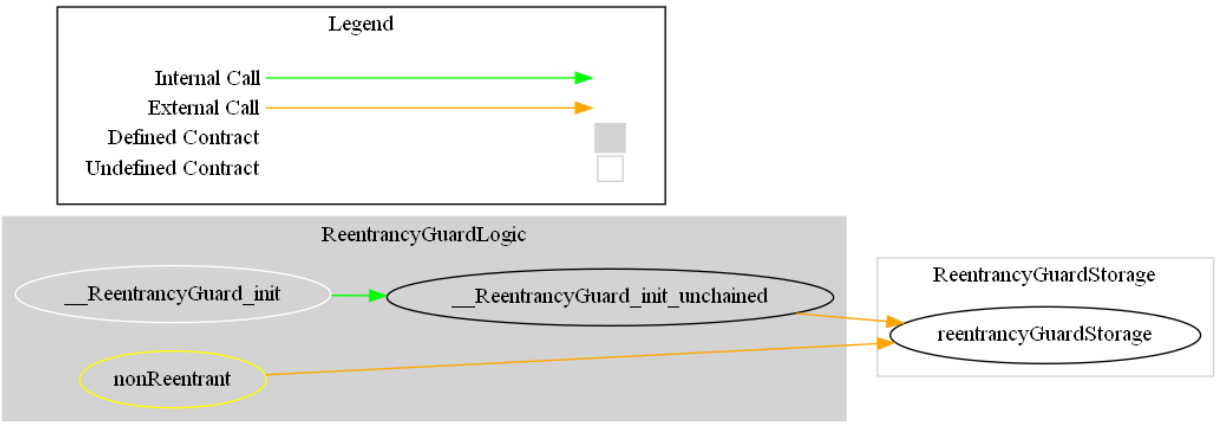
PermissionsEnumerableStorage



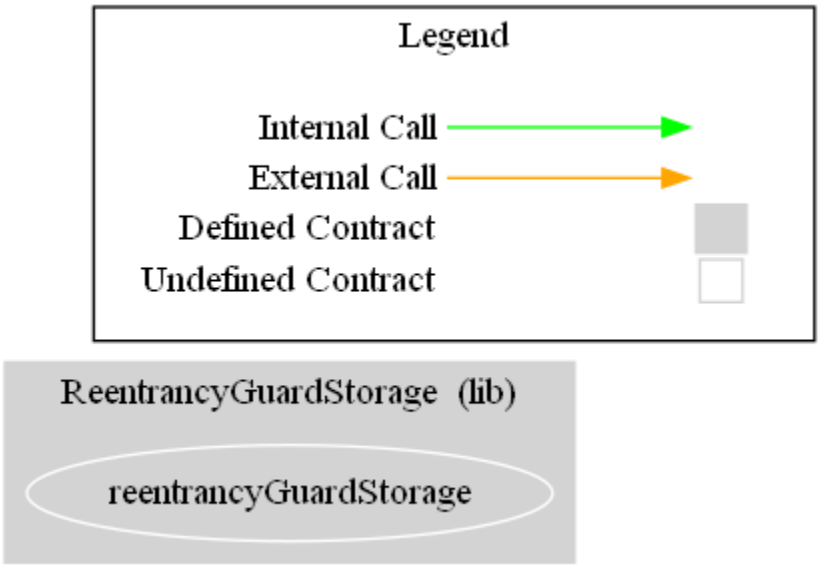
PluginMap



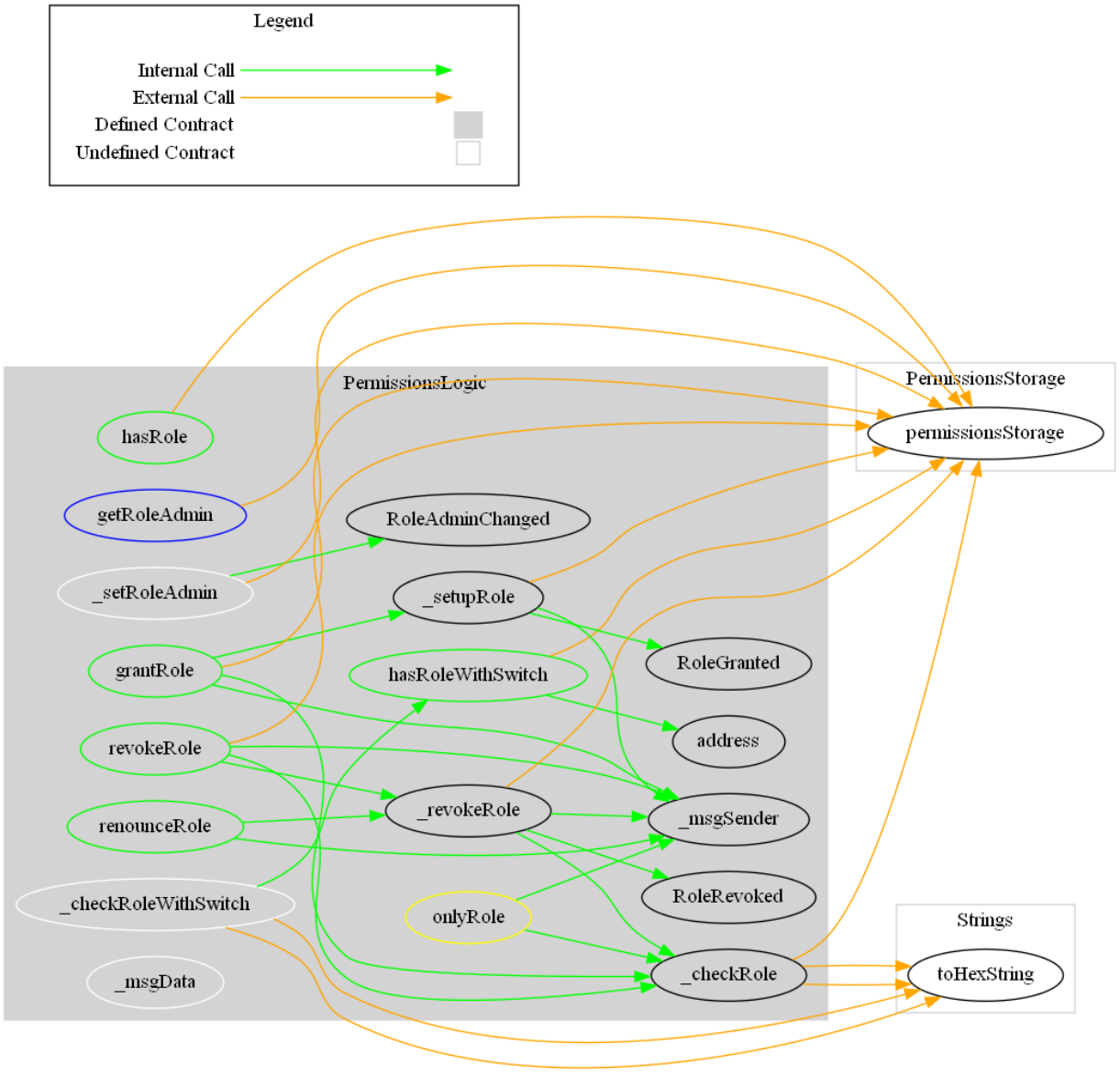
ReentrancyGuardLogic



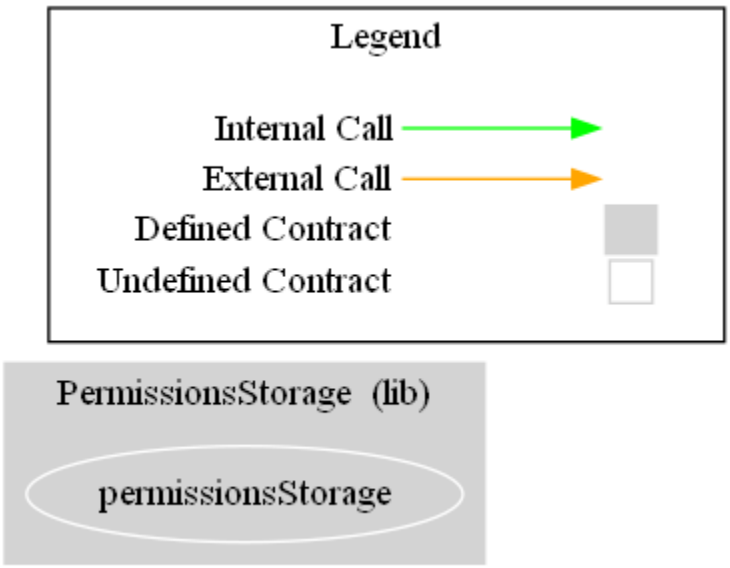
ReentrancyGuardStorage



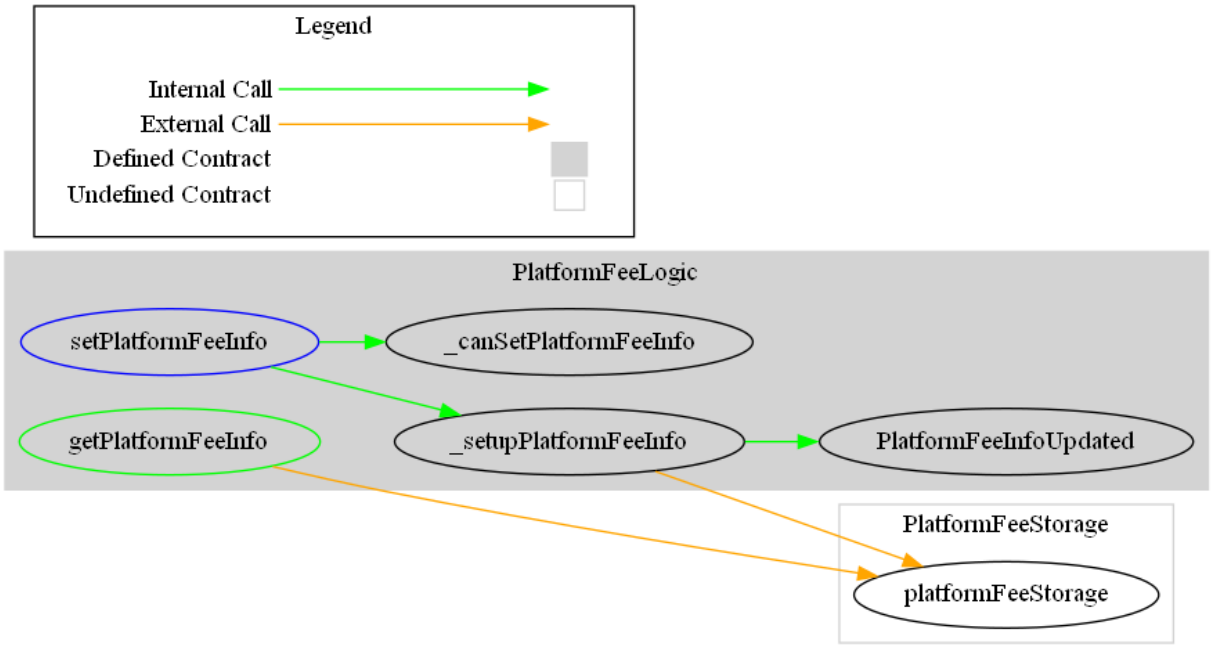
PermissionsLogic



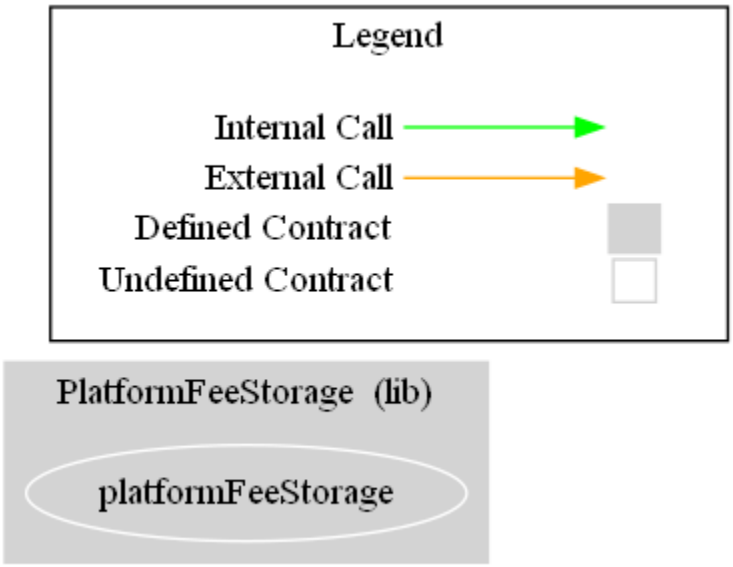
PermissionsStorage



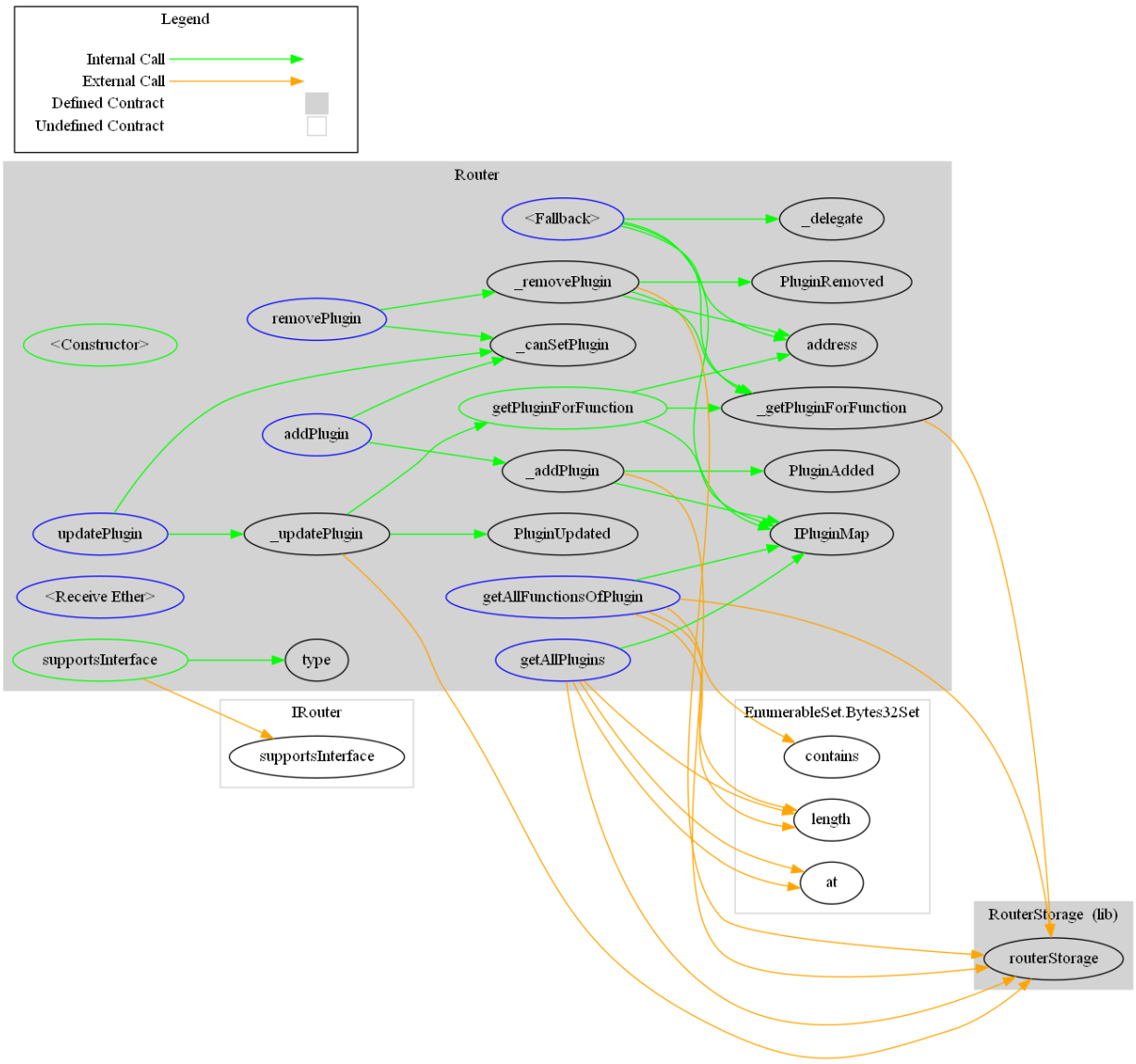
PlatformFeeLogic



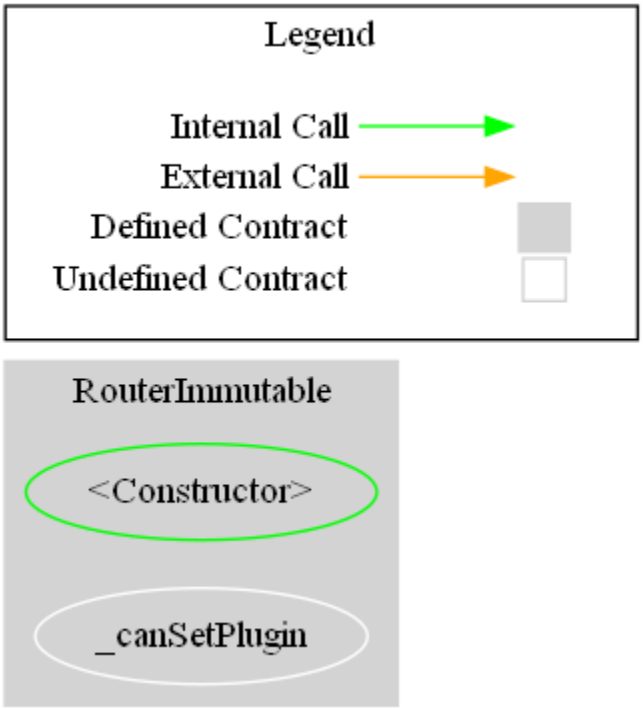
PlatformFeeStorage



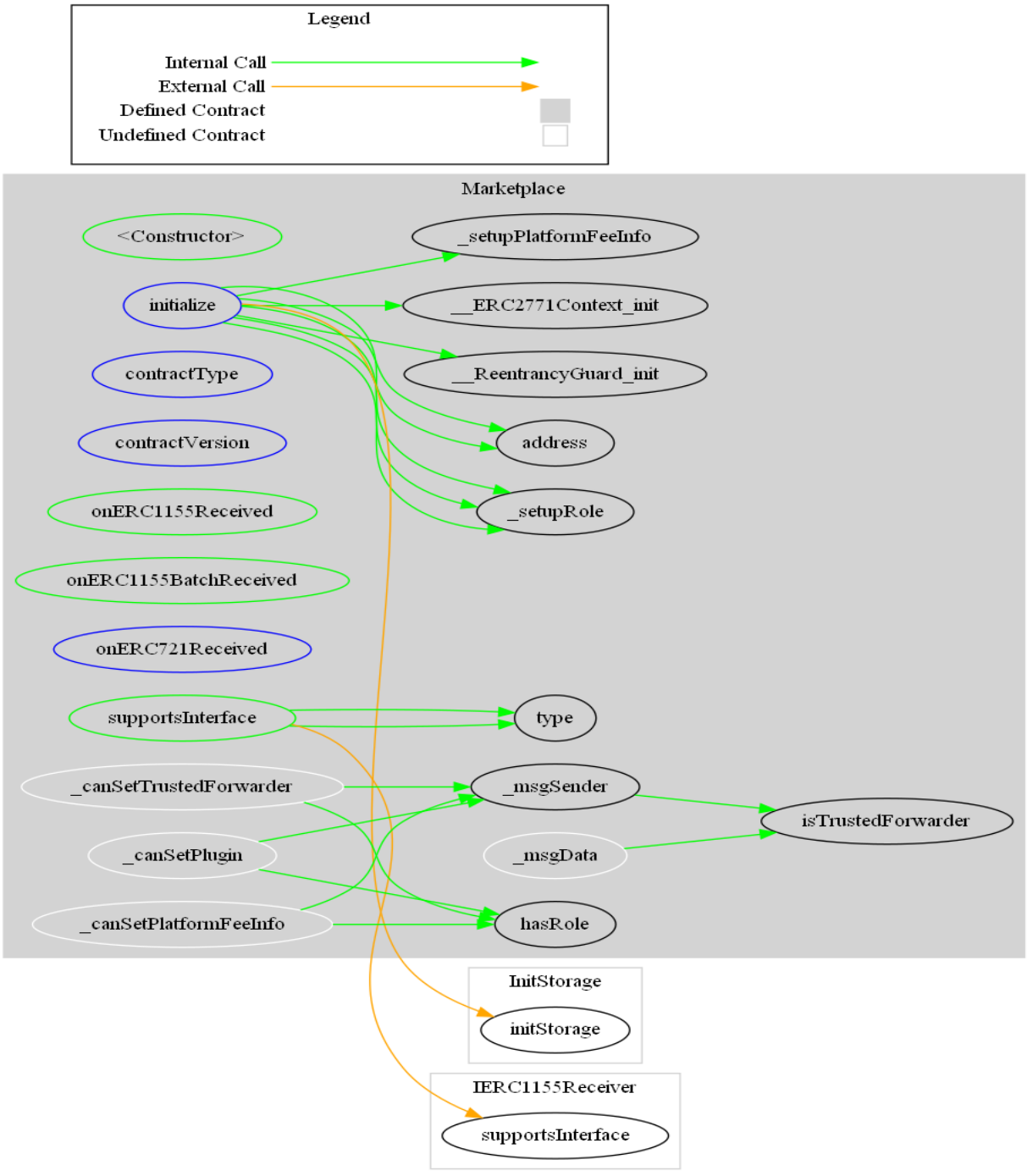
Router


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Router
<Fallback>
_delegate
_removePlugin
PluginRemoved
removePlugin
<Constructor>
_canSetPlugin
address
getPluginForFunction
_getPluginForFunction
addPlugin
_addPlugin
PluginAdded
updatePlugin
_updatePlugin
PluginUpdated
IPluginMap
<Receive Ether>
getAllFunctionsOfPlugin
supportsInterface
type
getAllPlugins
IRouter
supportsInterface
EnumerableSet.Bytes32Set
contains
length
at
RouterStorage (lib)
routerStorage

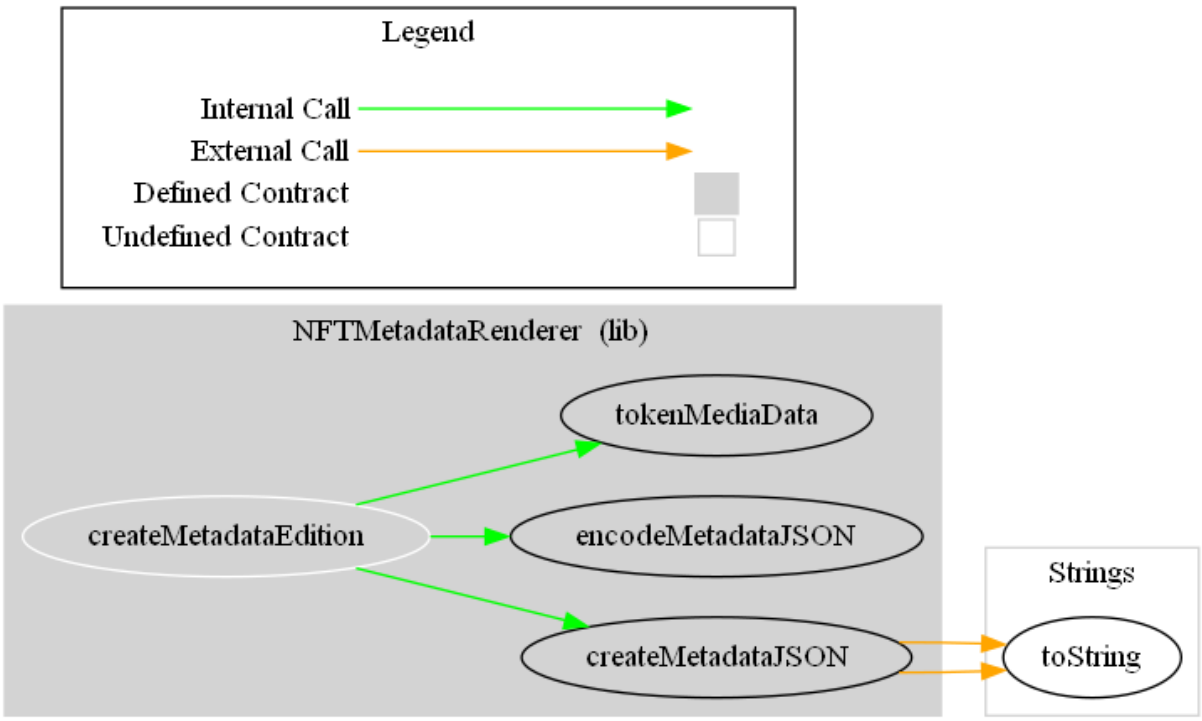
RouterImmutable



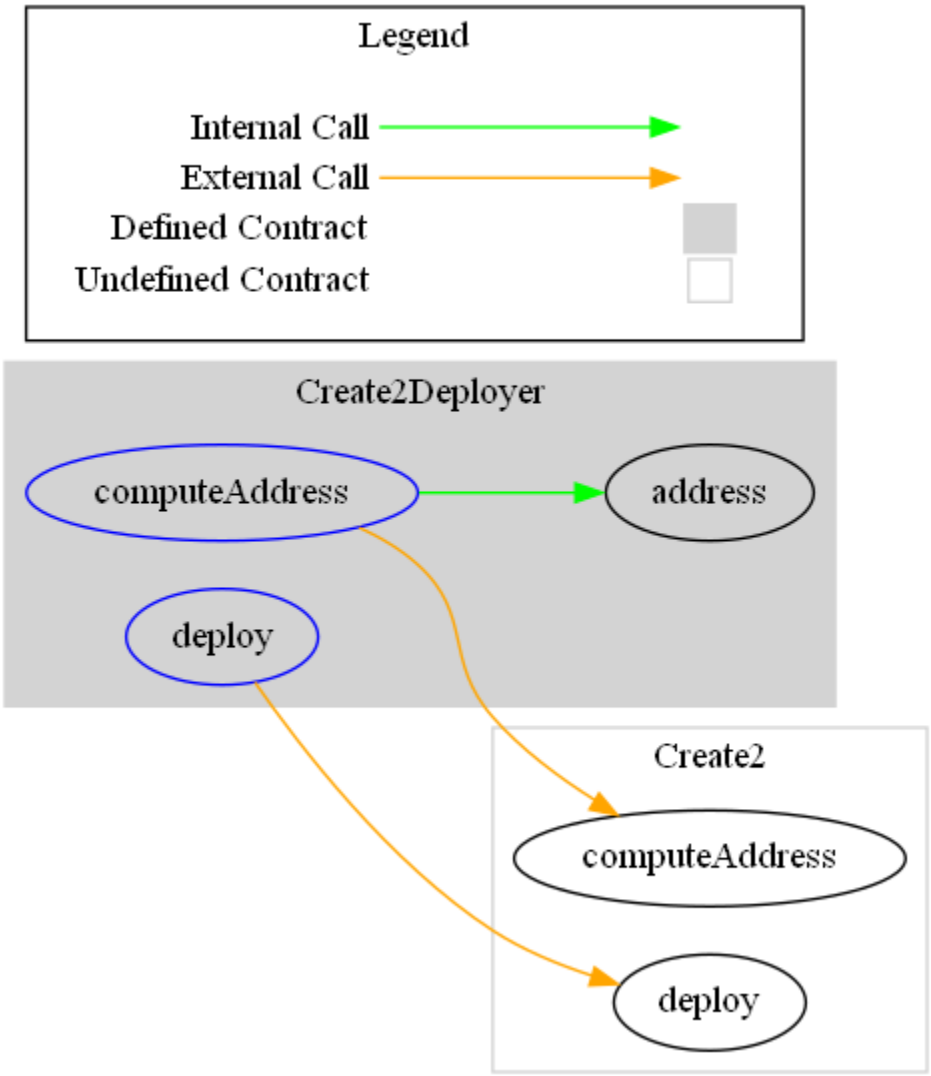
Marketplace



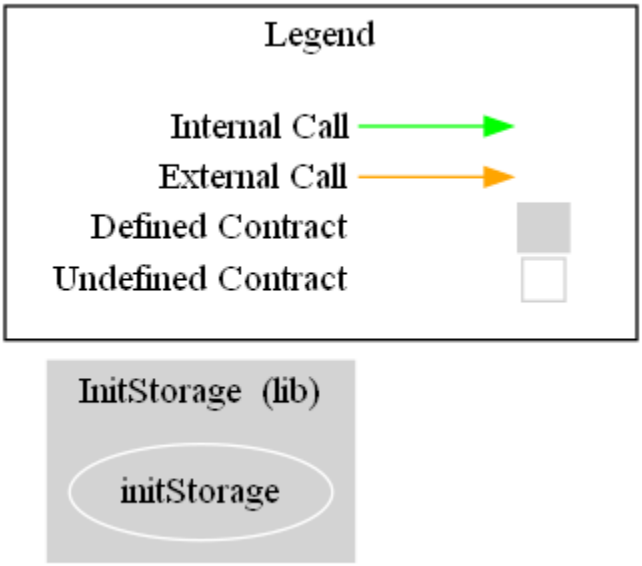
NFTMetadataRendererLib



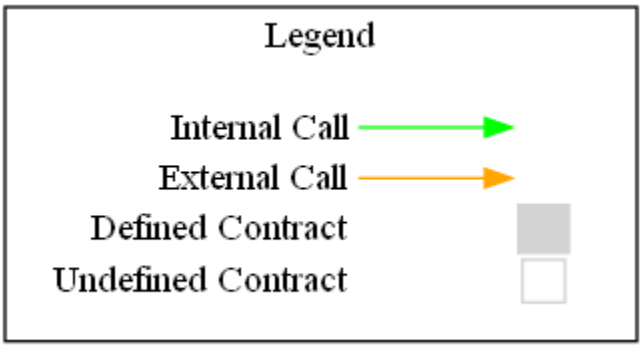
Create2Deployer



InitStorage

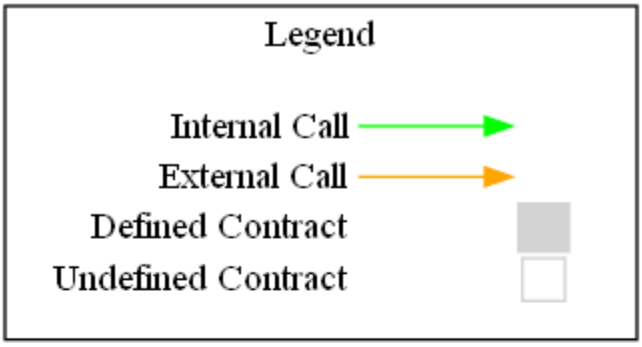


FeeType

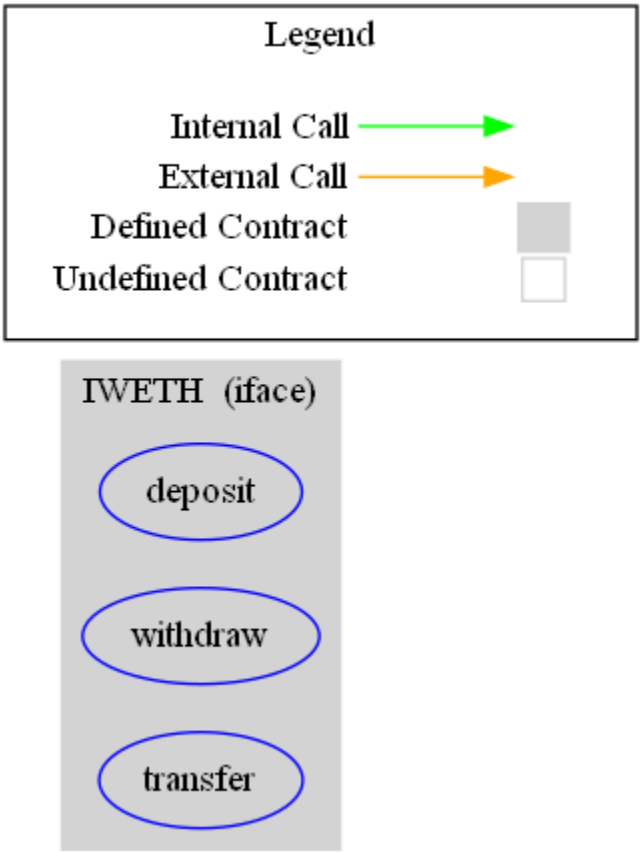


IMarketplace

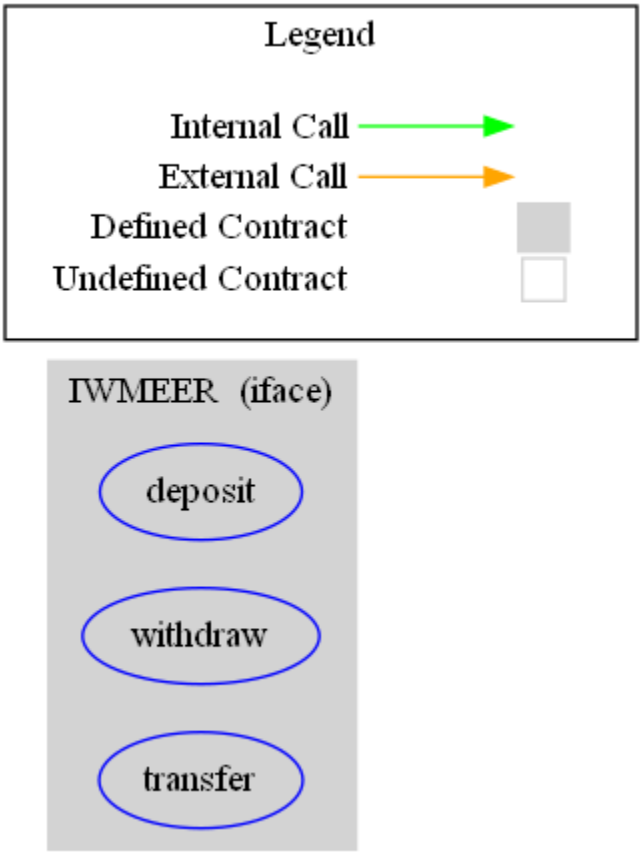
IPermissions



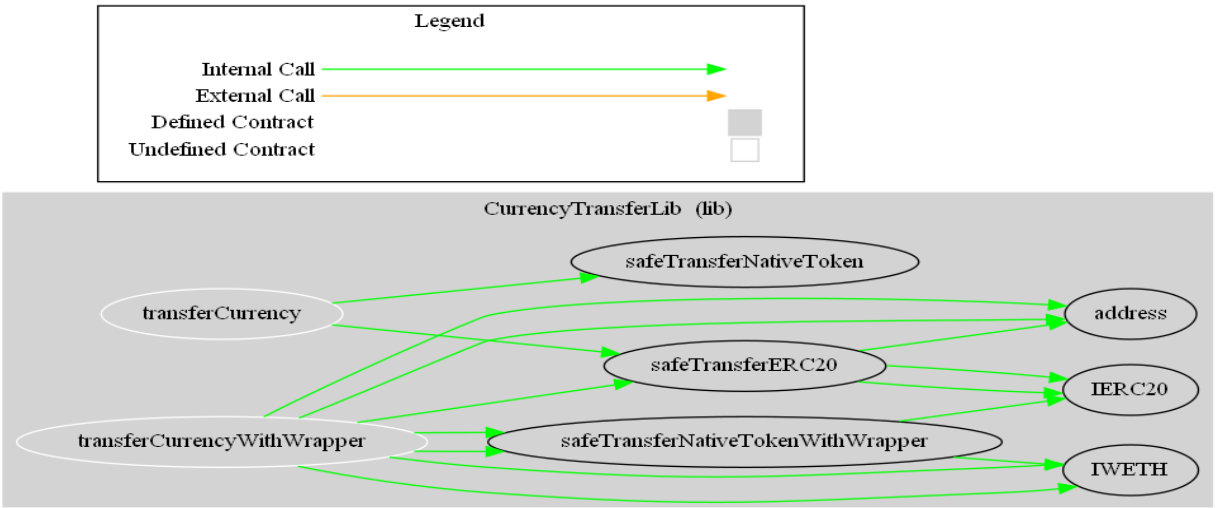
IWETH


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IWETH (iface)
deposit
withdraw
transfer

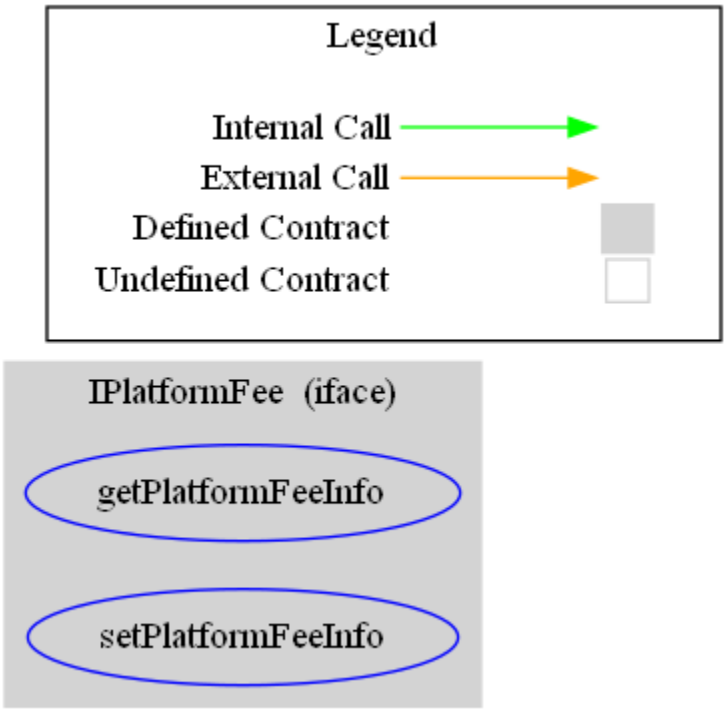
IWMEER



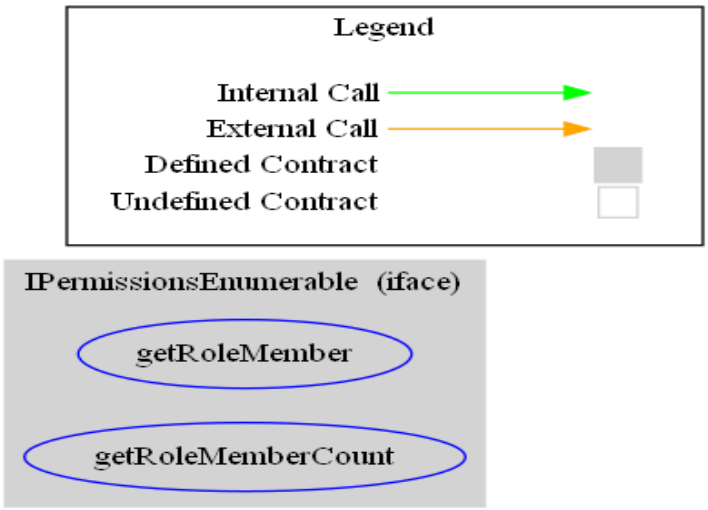
CurrencyTransferLib



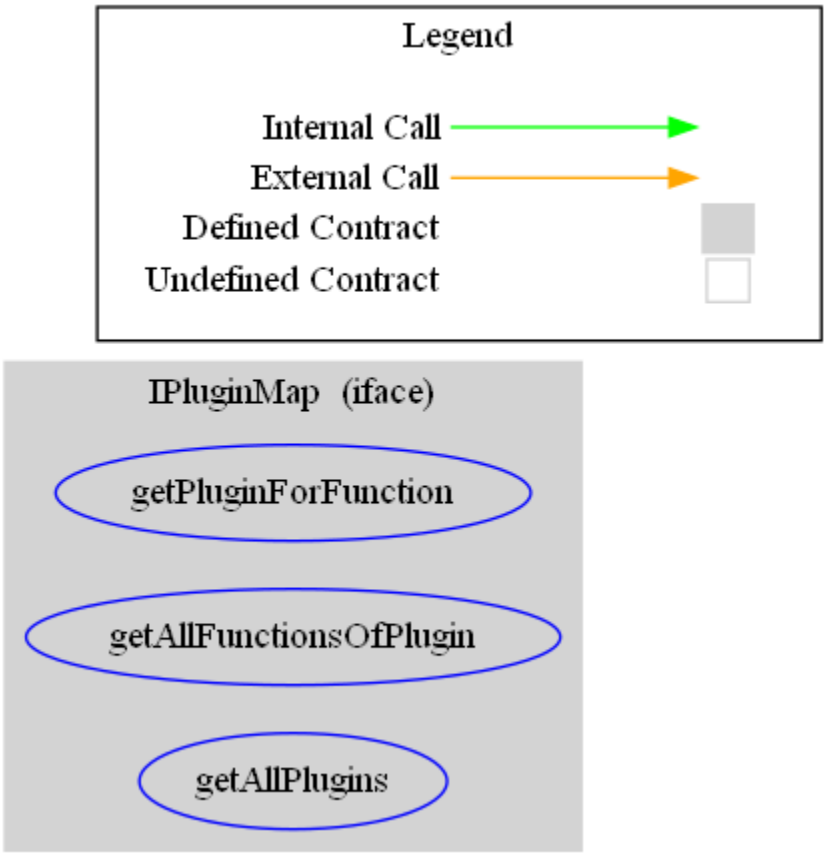
IPlatformFee



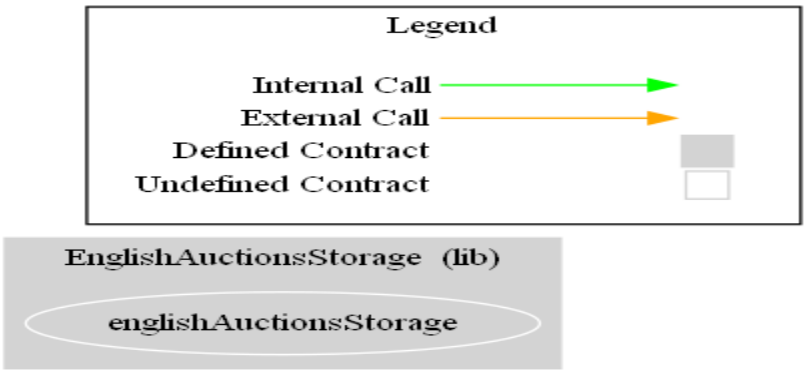
IPermissionsEnumerable



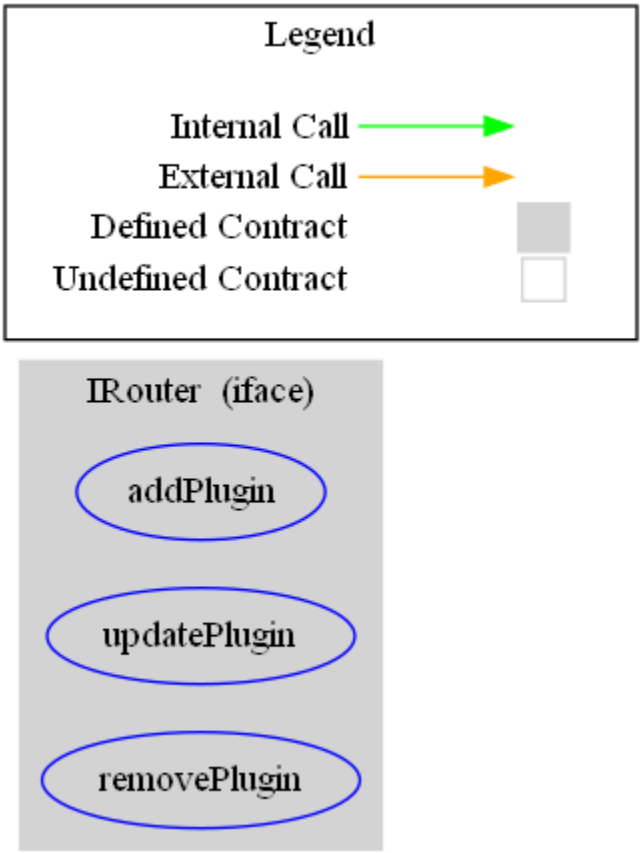
IPluginMap



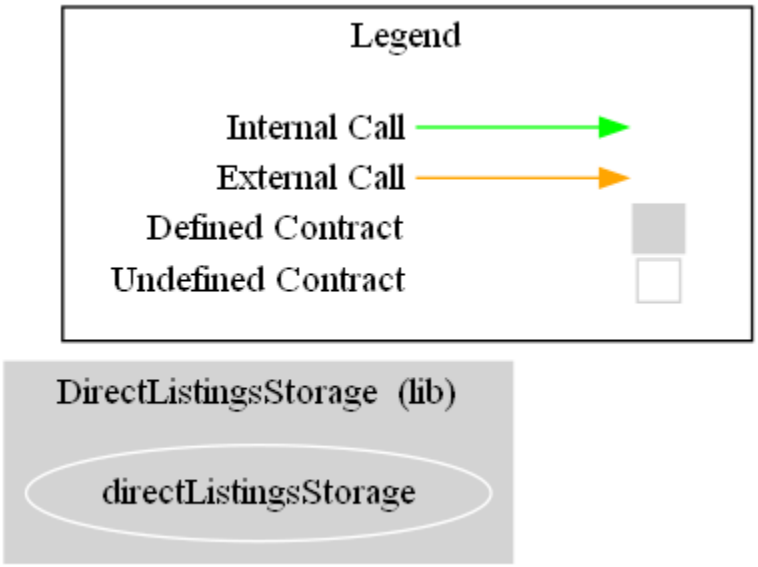
EnglishAuctionsStorage



IRouter


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IRouter (iface)
addPlugin
updatePlugin
removePlugin

DirectListingsStorage



Concluding Remarks

To wrap it up, this woowow audit has given us a good look at the contract's security and functionality.
We found some high,medium and low severity issues that need attention,we suggest taking action to address these findings.