HALBORN

Biconomy - Forwarder

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 1st, 2022 - April 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/07/2022	Omar Alshaeb
0.2	Draft Review	04/07/2022	Gabi Urrutia
1.0	Remediation Plan	04/12/2022	Omar Alshaeb
1.1	Remediation Plan Review	04/12/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Biconomy engaged Halborn to conduct a security audit on their smart contracts beginning on April 1st, 2022 and ending on April 8th, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were mostly addressed by the Biconomy team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

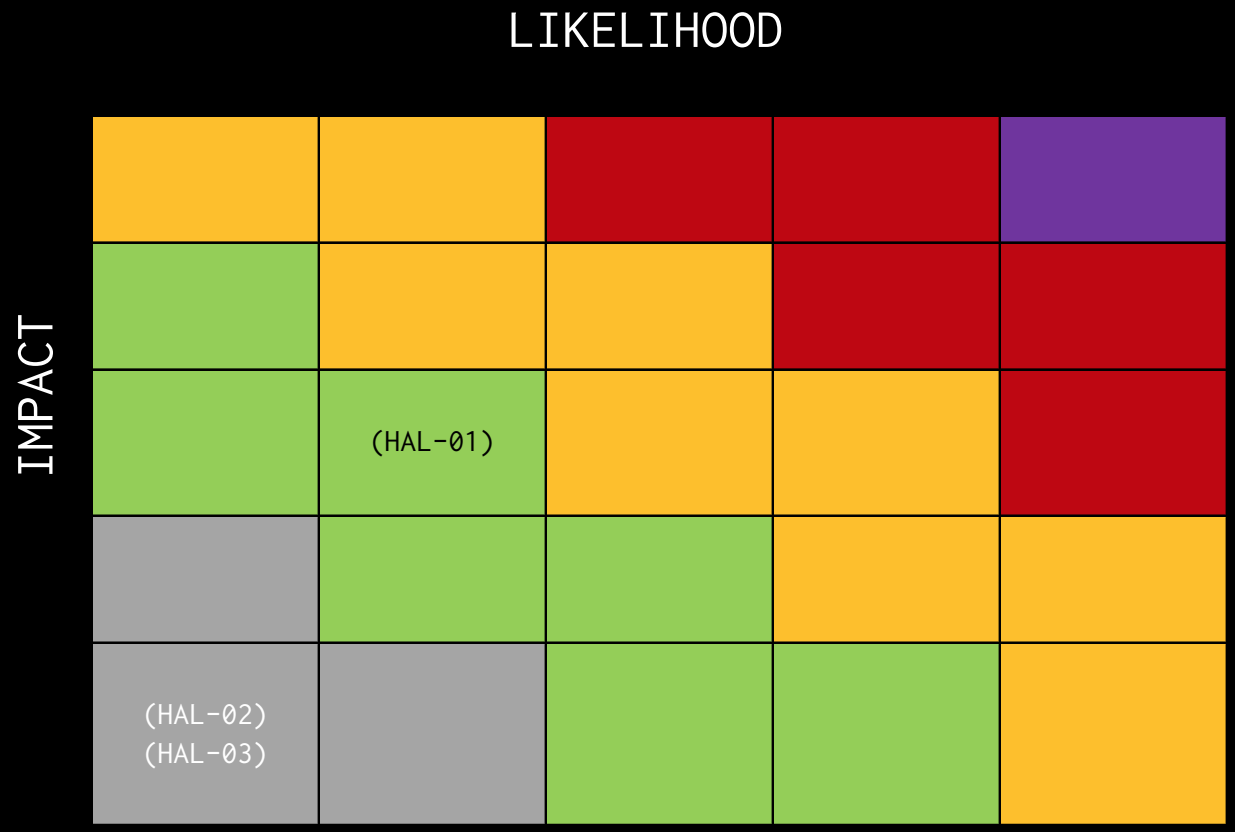
IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- ERC20ForwarderProxy.sol
- ERC20ForwarderImplementationV2.sol
- BiconomyForwarderV2.sol
- ForwardRequestTypesV2.sol
- FeeManager.sol
- OracleAggregator.sol

Commit ID: 611203c53e0b225f0d1c64e8b75adf9e2fc7dd29
Fixed Commit ID: 11f8abf0c0b8c26ce7a21972b44d1c91a7c3f756

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	2



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - OWNER CAN RENOUNCE OWNERSHIP	Low	RISK ACCEPTED
HAL02 - ZERO ADDRESS NOT CHECKED	Informational	SOLVED - 04/12/2022
HAL03 - MISSING EVENTS EMITTING	Informational	SOLVED - 04/12/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) OWNER CAN RENOUNCE OWNERSHIP - LOW

Description:

The Owner of the contract is usually the account that deploys the contract. As a result, the Owner can perform some privileged functions. In the ERC20ForwarderImplementationV2.sol, BiconomyForwarderV2.sol, FeeManager.sol and OracleAggregator.sol smart contracts, the renounceOwnership function is used to renounce the Owner permission. Renouncing ownership before transferring would result in the contract having no Owner, eliminating the ability to call privileged functions.

Code Location:

Listing 1: ERC20ForwarderImplementationV2.sol (Line 24)

```
24 contract ERC20ForwarderImplementationV2 is Initializable,
↳ OwnableUpgradeable, ForwardRequestTypesV2 {
25
```

Listing 2: BiconomyForwarderV2.sol (Line 23)

```
23 contract BiconomyForwarderV2 is ForwardRequestTypesV2, Ownable {
24
```

Listing 3: FeeManager.sol (Line 22)

```
22 contract FeeManager is IFeeManager, Ownable{
23
```

Listing 4: OracleAggregator.sol (Line 6)

```
6 contract OracleAggregator is Ownable{
7
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended that the Owner cannot call renounceOwnership without transferring the Ownership to other address first. In addition, if a multi-signature wallet is used, calling renounceOwnership function should be confirmed for two or more users.

Remediation Plan:

RISK ACCEPTED: The Biconomy team accepted the risk of this issue.

3.2 (HAL-02) ZERO ADDRESS NOT CHECKED - INFORMATIONAL

Description:

The function setTokenOracle within the contract OracleAggregator.sol is not verifying the callAddress parameter is not the zero address to avoid having issues retrieving the token price data feed.

Code Location:

Listing 5: OracleAggregator.sol (Line 22)

```
21     function setTokenOracle(address token, address callAddress,
↳ uint8 decimals, bytes calldata callData, bool signed) external
↳ onlyOwner{
22         tokensInfo[token].callAddress = callAddress;
23         tokensInfo[token].decimals = decimals;
24         tokensInfo[token].callData = callData;
25         tokensInfo[token].dataSigned = signed;
26     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

When setting an address variable, always ensure the value is not zero.

Remediation Plan:

SOLVED: The issue was solved in commit ID:
11f8abf0c0b8c26ce7a21972b44d1c91a7c3f756

3.3 (HAL-03) MISSING EVENTS EMITTING - INFORMATIONAL

Description:

It has been observed that critical functionality is missing emitting event for setFeeManager function within the ERC20ForwarderImplementationV2.sol contract and the setTokenAllowed function within the FeeManager.sol contract. These functions should emit events after completing the transactions.

Code Location:

Listing 6: ERC20ForwarderImplementationV2.sol

```
105     function setFeeManager(address _feeManager) external onlyOwner
 ↳ {
106         require(
107             _feeManager != address(0),
108             "ERC20Forwarder: new fee manager can not be a zero
 ↳ address"
109         );
110         feeManager = _feeManager;
111     }
```

Listing 7: FeeManager.sol

```
97     function setTokenAllowed(address token, bool allowed) external
 ↳  onlyOwner{
98         allowedTokens[token] = allowed;
99     }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Consider emitting an event when calling `setFeeManager` and `setTokenAllowed` functions.

Remediation Plan:

SOLVED: The issue was solved in commit ID: 11f8abf0c0b8c26ce7a21972b44d1c91a7c3f756

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

ERC20ForwarderProxy.sol

```
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ERC20ForwarderProxy.constructor(address,address)._implementation (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#28) shadows:
        - ERC1967Proxy._implementation() (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#29-31) (function)
        - Proxy._implementation() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#50) (function)
ERC20ForwarderProxy.constructor(address,address)._admin (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#28) shadows:
        - TransparentUpgradeableProxy._admin() (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#113-115) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Modifier TransparentUpgradeableProxy.ifAdmin() (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#45-51) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier

Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
        - Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#55)
                - _upgradeTo(newImplementation) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#105)
Reentrancy in TransparentUpgradeableProxy.constructor(address,address,bytes) (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#33-40):
        External calls:
        - ERC1967Proxy(_logic,_data) (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#37)
                - Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
        Event emitted after the call(s):
        - AdminChanged(_getAdmin(),newAdmin) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#142)
                - _changeAdmin(admin_) (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Proxy._delegate(address) (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#21-44) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#22-43)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
StorageSlot.getAddressSlot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#51-55) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#60-64) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#69-73) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#78-82) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#79-81)
ERC20ForwarderProxy.getAdmin() (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#32-37) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#34-36)
ERC20ForwarderProxy.getImplementation() (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#39-44) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#41-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#3)
        - ^0.8.2 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#3)
        - 0.8.4 (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
```

```
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
ERC1967Upgrade._getBeacon() (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#160-162) is never used and should be removed
ERC1967Upgrade._setBeacon(address) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#167-174) is never used and should be removed
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) is never used and should be removed
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) is never used and should be removed
Proxy._implementation() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#50) is never used and should be removed
StorageSlot.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#60-64) is never used and should be removed
StorageSlot.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#69-73) is never used and should be removed
StorageSlot.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#78-82) is never used and should be removed
TransparentUpgradeableProxy._admin() (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#113-115) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#3) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable ERC20ForwarderProxy.constructor(address,address)._implementation (contracts/forward-v2/forwarder/ERC20ForwarderProxy.sol#28) is too similar to TransparentUpgradeableProxy.implementation().implementation_ (
ransparentUpgradeableProxy.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

ERC20ForwarderImplementationV2.sol ForwardRequestTypesV2.sol

```
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

ERC20ForwarderImplementationV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes).transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#199) shadows:
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes).transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#227) shadows:
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.permitAndExecuteEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest).transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderIm
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.permitAndExecuteEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest).transferHandlerGas (contracts/forward-v2/forwarder/ERC20Forw
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.permitEIP2612AndExecuteEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest).transferHandlerGas (contracts/forward-v2/forwarder/ERC20Forw
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.permitEIP2612AndExecuteEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest).transferHandlerGas (contracts/forward-v2/forwarder/ER
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
ERC20ForwarderImplementationV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes).transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#385) shadows:
        - ERC20ForwarderImplementationV2.transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#27) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ECDSA.tryRecover(bytes32,bytes).r (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#59)' in ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.
d(uint256)(signature + 0x20) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in BiconomyForwarderV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#120-136):
        External calls:
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#130)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.from,msg.sender,req.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#135)
Reentrancy in ERC20ForwarderImplementationV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#189-202):
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#197)
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#200)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.token).transferFrom(req.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#414-417)
                - SafeERC20.safeTransferFrom(IERC20(req.token),req.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#420)
        External calls sending eth:
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#200)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.from,charge,req.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#201)
Reentrancy in BiconomyForwarderV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#148-164):
        External calls:
        - (success,ret) = req.request.to.call{gas: req.request.txGas}(abi.encodePacked(req.request.data,req.request.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#158)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.request.from,msg.sender,req.request.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#163)
Reentrancy in ERC20ForwarderImplementationV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#217-230):
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712Custom(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#225)
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#228)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.request.token).transferFrom(req.request.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#443-446)
                - SafeERC20.safeTransferFrom(IERC20(req.request.token),req.request.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#449)
        External calls sending eth:
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#228)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.request.from,charge,req.request.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#229)
Reentrancy in BiconomyForwarderV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#187-198):
        External calls:
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#192)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.from,msg.sender,req.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#197)
Reentrancy in ERC20ForwarderImplementationV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#376-388):
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executePersonalSign(req,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#383)
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#386)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
```

```
                - require(bool)(IERC20(req.token).transferFrom(req.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#414-417)
                - SafeERC20.safeTransferFrom(IERC20(req.token),req.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#420)
        External calls sending eth:
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#386)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.from,charge,req.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#387)
Reentrancy in ERC20ForwarderImplementationV2.permitAndExecuteEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest) (contracts/forward-v2/forwarder/ERC20ForwarderImpleme
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#256)
        - IERC20Permit(req.token).permit(permitOptions.holder,permitOptions.spender,permitOptions.nonce,permitOptions.expiry,permitOptions.allowed,permitOptions.v,permitOptions.r,permitOptions.s) (contracts/forward
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#261)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.token).transferFrom(req.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#414-417)
                - SafeERC20.safeTransferFrom(IERC20(req.token),req.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#420)
        External calls sending eth:
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#261)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.from,charge,req.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#262)
Reentrancy in ERC20ForwarderImplementationV2.permitAndExecuteEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest) (contracts/forward-v2/forwarder/ERC20Forwarder
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712Custom(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#289)
        - IERC20Permit(req.request.token).permit(permitOptions.holder,permitOptions.spender,permitOptions.nonce,permitOptions.expiry,permitOptions.allowed,permitOptions.v,permitOptions.r,permitOptions.s) (contracts
#291)
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#294)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.request.token).transferFrom(req.request.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#443-446)
                - SafeERC20.safeTransferFrom(IERC20(req.request.token),req.request.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#449)
        External calls sending eth:
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#294)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.request.from,charge,req.request.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#295)
Reentrancy in ERC20ForwarderImplementationV2.permitEIP2612AndExecuteEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest) (contracts/forward-v2/forwarder/ERC20Forwarder
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#322)
        - IERC20Permit(req.token).permit(permitOptions.holder,permitOptions.spender,permitOptions.value,permitOptions.expiry,permitOptions.v,permitOptions.r,permitOptions.s) (contracts/forward-v2/forwarder/ERC20For
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#327)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.token).transferFrom(req.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#414-417)
                - SafeERC20.safeTransferFrom(IERC20(req.token),req.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#420)
        External calls sending eth:
        - charge = _transferHandler(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#327)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.from,charge,req.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#328)
Reentrancy in ERC20ForwarderImplementationV2.permitEIP2612AndExecuteEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes,ForwardRequestTypesV2.PermitRequest) (contracts/forward-v2/forwarder/ERC20Fo
        External calls:
        - (success,ret) = BiconomyForwarderV2(forwarder).executeEIP712Custom(req,domainSeparator,sig) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#355)
        - IERC20Permit(req.request.token).permit(permitOptions.holder,permitOptions.spender,permitOptions.value,permitOptions.expiry,permitOptions.v,permitOptions.r,permitOptions.s) (contracts/forward-v2/forwarder/
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#360)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - require(bool)(IERC20(req.request.token).transferFrom(req.request.from,feeReceiver,charge)) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#443-446)
                - SafeERC20.safeTransferFrom(IERC20(req.request.token),req.request.from,feeReceiver,charge) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#449)
        External calls sending eth:
        - charge = _transferHandlerCustom(req,initialGas + baseGas + transferHandlerGas - postGas) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#360)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - FeeCharged(req.request.from,charge,req.request.token) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BiconomyForwarderV2._verifySigEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#231-253) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.deadline == 0 || block.timestamp + 20 <= req.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#243)
BiconomyForwarderV2._verifySigEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#266-288) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.request.deadline == 0 || block.timestamp + 20 <= req.request.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#278)
BiconomyForwarderV2._verifySigPersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#334-353) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.deadline == 0 || block.timestamp + 20 <= req.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#64-68)
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#119-122)
BiconomyForwarderV2.constructor() (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#43-50) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#46-48)
BiconomyForwarderV2.registerDomainSeparator(string,string) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#58-76) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#61-63)
BiconomyForwarderV2._verifySigEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#231-253) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#240-242)
BiconomyForwarderV2._verifySigEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#266-288) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#276-277)
BiconomyForwarderV2._verifyCallResult(bool,bytes,string) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#362-377) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#369-372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
        - 0.8.4 (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#2)
        - 0.8.4 (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#2)
        - 0.8.4 (contracts/forward-v2/forwarder/ForwardRequestTypesV2.sol#2)
        - 0.8.4 (contracts/forward-v2/forwarder/OracleAggregator.sol#2)
        - 0.8.4 (contracts/forward-v2/interfaces/IERC20Permit.sol#2)
        - 0.8.4 (contracts/forward-v2/interfaces/IFeeManager.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
ECDSA.recover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#131-139) is never used and should be removed
ECDSA.recover(bytes32,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#182-191) is never used and should be removed
ECDSA.toEthSignedMessageHash(bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#201-205) is never used and should be removed
ECDSA.toTypedDataHash(bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#216-218) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Low level call in BiconomyForwarderV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#120-136):
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#130)
Low level call in BiconomyForwarderV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#148-164):
        - (success,ret) = req.request.to.call{gas: req.request.txGas}(abi.encodePacked(req.request.data,req.request.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#158)
Low level call in BiconomyForwarderV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#187-198):
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#192)
Low level call in OracleAggregator._getTokenPrice(address) (contracts/forward-v2/forwarder/OracleAggregator.sol#36-44):
        - (success,ret) = tokensInfo[token].callAddress.staticcall(tokensInfo[token].callData) (contracts/forward-v2/forwarder/OracleAggregator.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.initialize(address,address,address)._feeReceiver (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#49) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.initialize(address,address,address)._feeManager (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#50) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.initialize(address,address,address)._forwarder (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#51) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.setTrustedForwarder(address)._forwarder (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#82) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.setFeeReceiver(address)._feeReceiver (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#94) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.setFeeManager(address)._feeManager (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#105) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.setTransferHandlerGas(address,uint256)._transferHandlerGas (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#147) is not in mixedCase
Parameter ERC20ForwarderImplementationV2.setSafeTransferRequired(address,bool)._safeTransferRequired (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#156) is not in mixedCase
Variable ERC20ForwarderImplementationV2._initializedVersion (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in ERC20ForwarderImplementationV2 (contracts/forward-v2/forwarder/ERC20ForwarderImplementat
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getNonce(address,uint256) should be declared external:
        - BiconomyForwarderV2.getNonce(address,uint256) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#88-92)
initialize(address,address,address) should be declared external:
        - ERC20ForwarderImplementationV2.initialize(address,address,address) (contracts/forward-v2/forwarder/ERC20ForwarderImplementationV2.sol#48-70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

BiconomyForwarderV2.sol ForwardRequestTypesV2.sol

```
Variable 'ECDSA.tryRecover(bytes32,bytes).r (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#59)' in ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.
d(uint256))(signature + 0x20) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in BiconomyForwarderV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#120-136):
        External calls:
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#130)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.from,msg.sender,req.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#135)
Reentrancy in BiconomyForwarderV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#148-164):
        External calls:
        - (success,ret) = req.request.to.call{gas: req.request.txGas}(abi.encodePacked(req.request.data,req.request.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#158)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.request.from,msg.sender,req.request.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#163)
Reentrancy in BiconomyForwarderV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#187-198):
        External calls:
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#192)
        Event emitted after the call(s):
        - MetaTransactionExecuted(req.from,msg.sender,req.data) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

BiconomyForwarderV2._verifySigEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#231-253) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.deadline == 0 || block.timestamp + 20 <= req.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#243)
BiconomyForwarderV2._verifySigEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#266-288) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.request.deadline == 0 || block.timestamp + 20 <= req.request.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#278)
BiconomyForwarderV2._verifySigPersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#334-353) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(req.deadline == 0 || block.timestamp + 20 <= req.deadline,request expired) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#66-68)
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#119-122)
BiconomyForwarderV2.constructor() (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#43-50) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#46-48)
BiconomyForwarderV2.registerDomainSeparator(string,string) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#58-76) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#61-63)
BiconomyForwarderV2._verifySigEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#231-253) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#248-252)
BiconomyForwarderV2._verifySigEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#266-288) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#276-277)
BiconomyForwarderV2._verifyCallResult(bool,bytes,string) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#362-377) uses assembly
        - INLINE ASM (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#369-372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
        - 0.8.4 (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#2)
        - 0.8.4 (contracts/forward-v2/forwarder/ForwardRequestTypesV2.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ECDSA.recover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#131-139) is never used and should be removed
ECDSA.recover(bytes32,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#162-191) is never used and should be removed
ECDSA.toEthSignedMessageHash(bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#201-205) is never used and should be removed
ECDSA.toTypedDataHash(bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#216-218) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in BiconomyForwarderV2.executeEIP712(ForwardRequestTypesV2.ERC20ForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#120-136):
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#130)
Low level call in BiconomyForwarderV2.executeEIP712Custom(ForwardRequestTypesV2.CustomForwardRequest,bytes32,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#148-164):
        - (success,ret) = req.request.to.call{gas: req.request.txGas}(abi.encodePacked(req.request.data,req.request.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#158)
Low level call in BiconomyForwarderV2.executePersonalSign(ForwardRequestTypesV2.ERC20ForwardRequest,bytes) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#187-198):
        - (success,ret) = req.to.call{gas: req.txGas}(abi.encodePacked(req.data,req.from)) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getNonce(address,uint256) should be declared external:
        - BiconomyForwarderV2.getNonce(address,uint256) (contracts/forward-v2/forwarder/BiconomyForwarderV2.sol#88-92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

FeeManager.sol

```
FeeManager.setDefaultFeeMultiplier(uint16) (contracts/forward-v2/feeManager/FeeManager.sol#65-67) should emit an event for:
        - bp = _bp (contracts/forward-v2/feeManager/FeeManager.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.4 (contracts/forward-v2/feeManager/FeeManager.sol#2)
        - 0.8.4 (contracts/forward-v2/interfaces/IFeeManager.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter FeeManager.setDefaultFeeMultiplier(uint16)._bp (contracts/forward-v2/feeManager/FeeManager.sol#65) is not in mixedCase
Parameter FeeManager.setDefaultTokenFeeMultiplier(address,uint16)._bp (contracts/forward-v2/feeManager/FeeManager.sol#69) is not in mixedCase
Parameter FeeManager.setUserTokenFeeMultiplier(address,address,uint16)._bp (contracts/forward-v2/feeManager/FeeManager.sol#81) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

OracleAggregator.sol

```
Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.4 (contracts/forward-v2/forwarder/OracleAggregator.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in OracleAggregator._getTokenPrice(address) (contracts/forward-v2/forwarder/OracleAggregator.sol#36-44):
        - (success,ret) = tokensInfo[token].callAddress.staticcall(tokensInfo[token].callData) (contracts/forward-v2/forwarder/OracleAggregator.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives. The actual vulnerabilities found by Slither are already included in the report findings.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

BiconomyForwarderV2.sol

Line	SWC Title	Severity	Short Description
28	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
41	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
243	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
340	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

OracleAggregator.sol

Line	SWC Title	Severity	Short Description
37	(SWC-104) Unchecked Call Return Value	Medium	The return value of a message call is not checked.

FeeManager.sol

Line	SWC Title	Severity	Short Description
24	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
26	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
28	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
30	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
32	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
34	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
36	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

- No major issues were found by Mythx.

THANK YOU FOR CHOOSING

// HALBORN