Biconomy

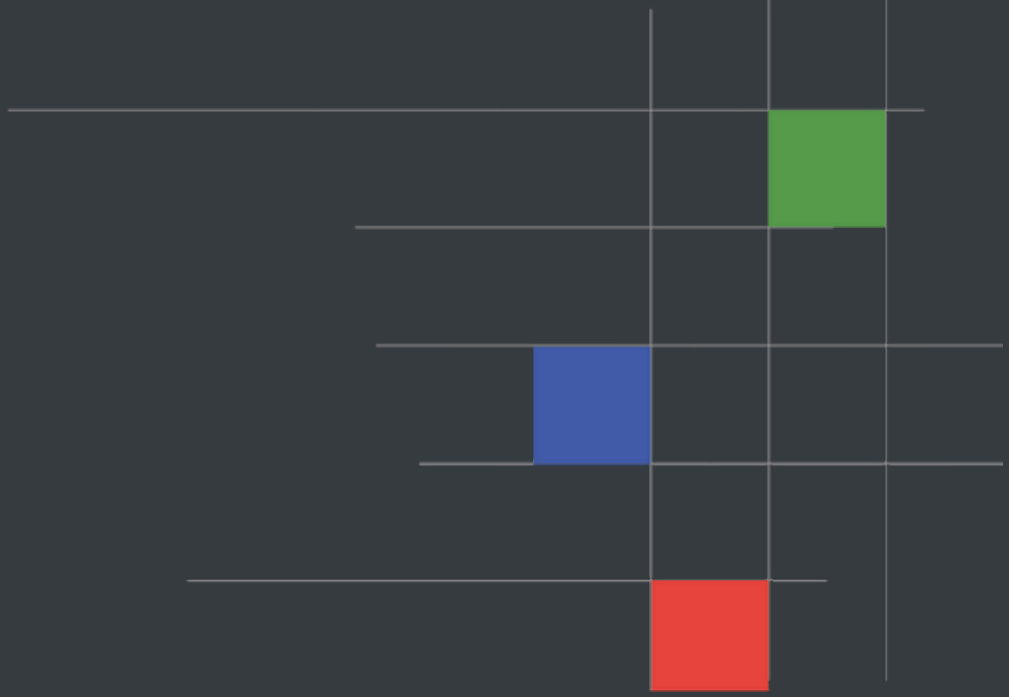
Smart Contracts

Security Assessment

March 17th, 2021

By:
Sheraz Arshad @ CertiK
sheraz.arshad@certik.org

Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org



Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Biconomy
Description	The audited codebase comprise of contracts allowing relaying of transaction containing messages signed by users and receiving payments in the form of ERC20 token contracts. Other contracts include managing liquidity pool and implementation of access rights for Executor.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 6233e44a5701a202390bbd2cd391d8b8699e234a 2. f785038d30530ade41dbba93f93af78a980329c1

Audit Summary

Delivery Date	March 17th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	February 16th, 2021 - March 17th, 2021

Vulnerability Summary

Total Issues	41
Total Critical	0
Total Major	2
Total Medium	9
Total Minor	13
Total Informational	17

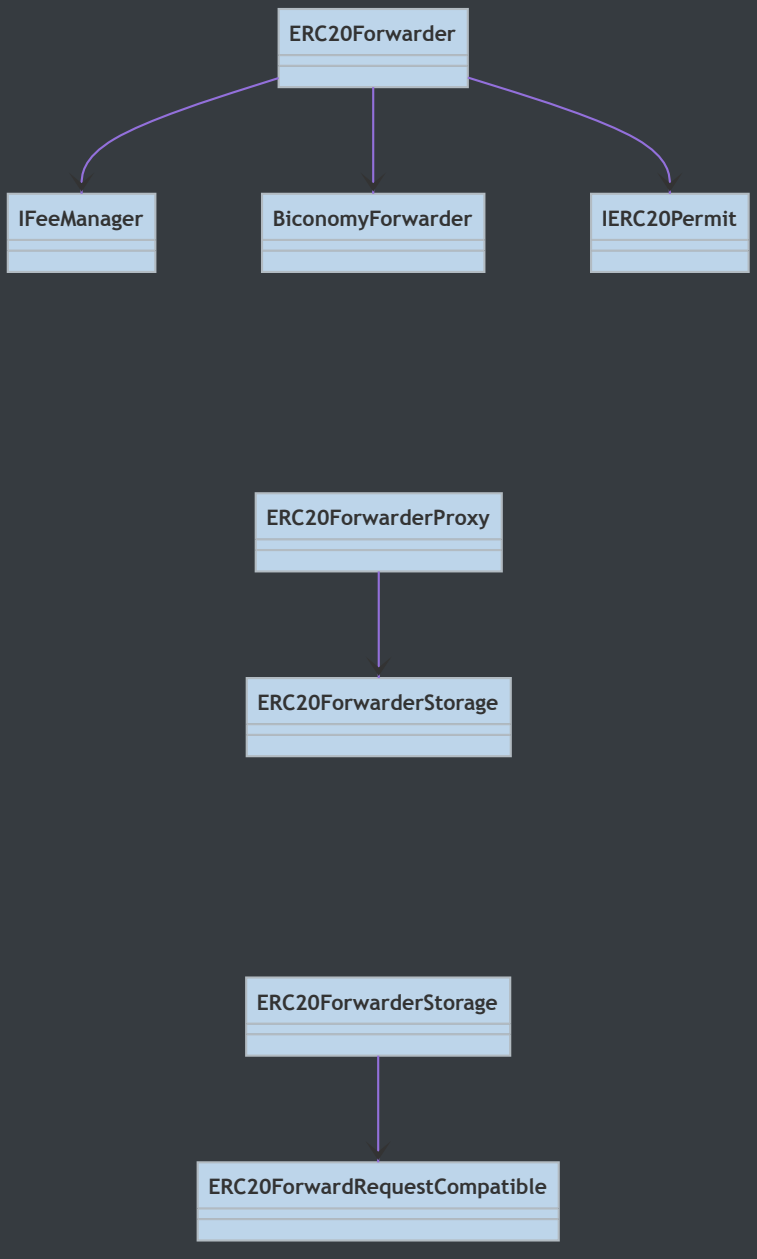
Executive Summary

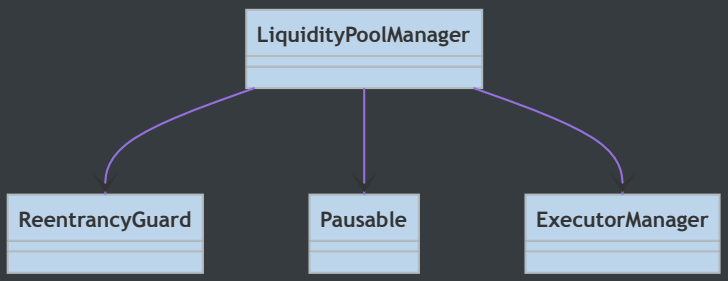
This report represents the results of CertiK's engagement with Biconomy on their implementation of the Meta transaction protocol contracts.
The codebase comprise Forwarder contracts for the implementation of meta transaction, Liquidity Pool manager contract and Executor Manager contract. Static analysis and manual inspection was performed on the contracts. Our findings include 2 major issues with some medium and minor issues and the rest of the issues are related to code optimizations. All of the findings were remediated by the Biconomy team.

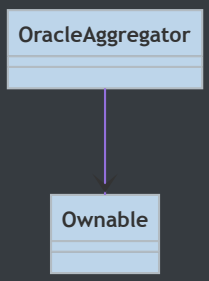
Files In Scope

ID	Contract	Location
BFR	BiconomyForwarder.sol	contracts/6/forwarder/BiconomyForwarder.sol
ERC	ERC20Forwarder.sol	contracts/6/forwarder/ERC20Forwarder.sol
EMR	ExecutorManager.sol	contracts/6/ExecutorManager.sol
ERF	ERC20ForwarderProxy.sol	contracts/6/forwarder/ERC20ForwarderProxy.sol
ERT	ERC20TransferHandler.sol	contracts/6/forwarder/ERC20TransferHandler.sol
ERS	ERC20ForwarderStorage.sol	contracts/6/forwarder/ERC20ForwarderStorage.sol
ERR	ERC20ForwardRequestCompatible.sol	contracts/6/forwarder/ERC20ForwardRequestCompatible.sol
LPM	LiquidityPoolManager.sol	contracts/6/insta-swaps/LiquidityPoolManager.sol
OAR	OracleAggregator.sol	contracts/6/forwarder/OracleAggregator.sol

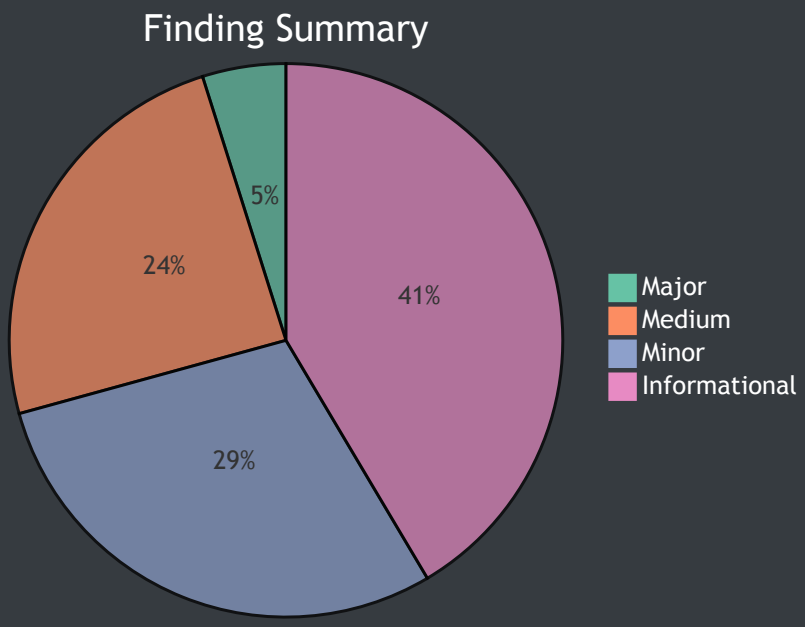
File Dependency Graph




LiquidityPoolManager
ReentrancyGuard
Pausable
ExecutorManager


OracleAggregator
Ownable

Findings



ID	Title	Type	Severity	Resolved
BFR-01	Order of variables for `EIP712` is not followed	Logical Issue	Medium	✓
BFR-02	Function Visibility Optimization	Gas Optimization	Informational	✓
BFR-03	Return value from the function is never used	Control Flow	Informational	✓
BFR-04	Functions are incorrectly marked as `payable`	Logical Issue	Medium	✓
BFR-05	Plain ether received by the contract are incorrectly transferred to message signer from subsequent	Logical Issue	Medium	✓

	transaction of any of `execute` functions			
BFR-06	Possibility of replay attack in the event of a fork	Logical Issue	Medium	✓
ERC-01	Order of contracts inheritance does not allow storage upgradeability of `ERC20ForwarderStorage`	Logical Issue	Minor	✓
ERC-02	Lack of verification for the passed function argument	Logical Issue	Minor	✓
ERC-03	No event emitted for external state variable change	Data Flow	Minor	✓
ERC-04	Lack of verification for the passed function argument	Logical Issue	Minor	✓
ERC-05	Lack of verification for the passed function argument	Logical Issue	Minor	✓
ERC-06	Lack of verification for the passed function argument	Logical Issue	Minor	✓
ERC-07	Lack of verification for the passed function argument	Logical Issue	Minor	✓
ERC-08	Explicitly returning local variable	Gas Optimization	Informational	✓
ERC-09	Function Visibility Optimization	Gas Optimization	Informational	✓
ERC-10	Possibility of ether being trapped in the contract	Logical Issue	Major	✓
ERC-11	Ineffectual declaration of utilization of `SafeMath` library for `uint128`	Volatile Code	Informational	✓
ERR-01	The names of file and contract are different	Language Specific	Informational	✓
ERF-01	Missing reason string	Coding Style	Informational	✓
ERF-02	Lack of verification for the passed constructor argument	Logical Issue	Medium	✓
ERF-	Function Visibility Optimization	Gas		✓

03		Optimization	Informational	
ERF-04	Order of contracts inheritance does not allow storage upgradeability of `ERC20ForwarderStorage`	Logical Issue	Minor	✓
ERF-05	Lack of verification for the constructor argument	Logical Issue	Medium	✓
ERS-01	Type `address payable` can be changed to `address`	Language Specific	Informational	✓
OAR-01	mappings data can be packed in a struct	Gas Optimization	Informational	✓
OAR-02	Missing state variable's visibility	Language Specific	Informational	✓
OAR-03	Function Visibility Optimization	Gas Optimization	Informational	✓
EMR-01	Lack of verification of the constructor argument	Logical Issue	Medium	✓
EMR-02	Function Visibility Optimization	Gas Optimization	Informational	✓
LPM-01	Unspecified state variable visibility	Language Specific	Informational	✓
LPM-02	mappings data can be packed in a struct	Gas Optimization	Informational	✓
LPM-03	Lack of verification for the constructor parameters	Logical Issue	Medium	✓
LPM-04	Mutability Specifiers Missing	Gas Optimization	Informational	✓
LPM-05	`LiquidityAdded` event is not emitted	Volatile Code	Minor	✓
LPM-06	Use of `send` to transfer ether	Volatile Code	Minor	✓
LPM-07	`LiquidityRemoved` event is not emitted	Volatile Code	Minor	✓
LPM-08	Usage of `msg.sender` instead of `_msgSender()`	Logical Issue	Major	✓

LPM-09	Redundant code	Gas Optimization	Informational	✓
LPM-10	Incorrect comparison	Logical Issue	Minor	✓
LPM-11	Possibility of re-entrancy due to the use of `send` for ether transfer	Volatile Code	Minor	✓
LPM-12	Contract receives plain ether with no side-effects	Volatile Code	Medium	✓

BFR-01: Order of variables for `EIP712` is not followed

Type	Severity	Location
Logical Issue	Medium	BiconomyForwarder.sol L28

Description:

The aforementioned line declares the domain type for `EIP712`. The domain type contains variable `salt` of type `uint256` which is incorrect as the variable stores the `chainid` instead of salt. It can lead to incorrect signing of message if a random `uint256` value is used as salt when signing the message off-chain. Additionally, the current domain hash is incompatible with the `EIP712` standard as the standard denotes a strict order of variables and permits them to be omitted only

Recommendation:

We advise to rename the `salt` variable name on the aforementioned line to `chainid` to comply with the standard of declaring domain type accordingly with the `EIP712`.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The `chaiid` was omitted from the signature and salt representing chain id is added with the Biconomy team stating "chainId to salt is done to support our network agnostic transactions as wallets will refuse to sign if the chainId doesn't match the currently selected network".

BFR-02: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	BiconomyForwarder.sol L90, L108, L130, L146

Description:

The functions on the aforementioned line have `external` visibilities yet some of their parameters have `memory` as data location. It is gas efficient to have `calldata` as memory location for reference type variables if the function's visibility is `external`

Recommendation:

We advise that the parameters with `memory` data location have their data location changed to `calldata`

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

BFR-03: Return value from the function is never used

Type	Severity	Location
Control Flow	Informational	BiconomyForwarder.sol L246

Description:

The function on the aforementioned line either returns its parameter `returndata` or reverts the transaction. The returned value from the function is never utilized in any of the instances where it is called.

Recommendation:

We advise to remove the return type from the function signature as the returned value is never utilized.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

BFR-04: Functions are incorrectly marked as `payable`

Type	Severity	Location
Logical Issue	Medium	BiconomyForwarder.sol L108, L146

Description:

The functions on the aforementioned lines are incorrectly marked as `payable` as any ether sent along with the function call will be forwarded to the message signer of the transaction.

Recommendation:

We advise to remove the `payable` modifier from the signatures of the aforementioned functions so that they do not accept ether.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

BFR-05: Plain ether received by the contract are incorrectly transferred to message signer from subsequent transaction of any of `execute` functions

Type	Severity	Location
Logical Issue	Medium	BiconomyForwarder.sol L69, L120-L122, L154-L156

Description:

The `receive` function on `L69` will allow the receiving of plain ethers (without involving function call from the contract itself) but they would be sent to the message signer from the subsequent transaction involving any of `execute` functions.
The lines `L120-L122` and `L154-L156` route any ether received by the forwarder to the message signer. The forwarder contract should not be allowed to receive ether and it should be the job of the recipient contract to make sure the funds are routed to message signer by utilizing `_msgSender()` instead of `msg.sender` as it would be considered an unsafe implementation in the recipient contract.

Recommendation:

The forwarder contract should not deal with ethers, it should `revert` sending of ethers because the recipient contract should take the responsibility of sending ether to message signer (recipient) and this code part can be removed along with the `receive` function. The sending of ethers to `msg.sender` in the recipient contract is a bug itself because the contract should deal the message signer as transaction sender and not the forwarder contract.

If the current functionality must be kept then a lock variable in the contract state can be introduced which is only set to `true` at the start of function execution and allows receiving of ether in the `receive` function and then set back to `false` at the end of function execution so the arbitrary plain ether could not be received in between the contract's function calls.

The locking mechanism of ether receiving can be referenced from the following code example:

```
contract Example {
    bool ethLocked = true;

    receive() external payable {
        require(!ethLocked, "eth is locked");
    }

    function execute() external {
        ethLocked = false;
        // perform execution of functionality
        ethLocked = true;
    }
}
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The `receive` function is removed from the contract.

BFR-06: Possibility of replay attack in the event of a fork

Type	Severity	Location
Logical Issue	Medium	BiconomyForwarder.sol L46

Description:

The function the aforementioned line registers domains by getting `chainid` from the context of the function. The functions verifying the signatures only check that if the domain is registered. A possible replay attack can be performed in the event of a fork where signatures from the chain will be executable on the fork as there is no check performed in the function verifying the signatures if the chainid in the context of the function matches the chainid in the domain hash.

Recommendation:

We advise that the `chainid` is cached in the storage of contract upon construction and checked against on each invocation of signature validation and if the cached one differs so that the signatures could not be replayed on the fork..

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-01: Order of contracts inheritance does not allow storage upgradeability of `ERC20ForwarderStorage`

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L27

Description:

The order of contracts inheritance does not allow storage upgradeability if new storage variables are needed to be introduced in `ERC20ForwarderStorage` because any new storage variable introduced in the `ERC20ForwarderStorage` will overwrite the storage slot currently referenced by the variable `_owner`.

Recommendation:

We advise to to place `ERC20ForwarderStorage` after the `Ownable` contract in the inheritance chain so the storage of `ERC20Forwarder` contract could be updated or expanded by modifying `ERC20ForwarderStorage` contract.

```
    contract ERC20Forwarder is Ownable, ERC20ForwarderStorage {}
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-02: Lack of verification for the passed function argument

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L72

Description:

The function parameter `oa` on the aforementioned line is not validated against zero value.

Recommendation:

We advise to validate the function parameter `oa` of type `address` against zero value.

```
    require(oa != address(0), "oracleAggregator cannot be a zero address");
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

ERC-03: No event emitted for external state variable change

Type	Severity	Location
Data Flow	Minor	ERC20Forwarder.sol L72

Description:

The external `setOracleAggregator` function in the assigns the supplied `oa` parameter to the public `oracleAggregator` state variable variable without emitting an event, which makes it difficult to track off-chain.

Recommendation:

Consider creating and emitting an event in order to track when the state of the `oracleAggregator` variable changes.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The event `OracleAggregatorChanged` is emitted in the function's body.

ERC-04: Lack of verification for the passed function argument

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L77

Description:

The function parameter `_forwarder` is not validated against zero value.

Recommendation:

We advise to perform the check against zero value for the parameter `_forwarder` of type `address`.

```
require(
   _forwarder != address(0),
   "_forwarder cannot be zero"
);
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-05: Lack of verification for the passed function argument

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L123

Description:

The parameter `_feeReceiver` on the aforementioned line is not checked against zero value.

Recommendation:

We advise to perform the check against zero value for parameter `_feeReceiver` of type `address`.

```
require(
   _feeReceiver != address(0),
   "_feeReceiver cannot be zero"
);
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-06: Lack of verification for the passed function argument

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L131

Description:

The parameter `_feeManager` on the aforementioned line is not validated against zero value.

Recommendation:

We advise to perform the check against zero value for parameter `_feeManager` of type `address`.

```
require(
   _feeManager != address(0),
   "_feeManager cannot be zero"
);
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-07: Lack of verification for the passed function argument

Type	Severity	Location
Logical Issue	Minor	ERC20Forwarder.sol L144, L149

Description:

The function parameters of `token` on the aforementioned lines is not validated against zero value.

Recommendation:

We advise to perform check against zero value for the parameter `token` of type `address` on the aforementioned lines.

```
    require(
       token != address(0),
       "token cannot be zero"
    );
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-08: Explicitly returning local variable

Type	Severity	Location
Gas Optimization	Informational	ERC20Forwarder.sol L160

Description:

The function on the aforementioned line explicitly returns a local variable which increases the overall cost of gas.

Recommendation:

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-09: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	ERC20Forwarder.sol L181, L211, L244, L277, L310, L345, L375, L403

Description:

The functions on the aforementioned line have `external` visibilities yet some of their parameters have `memory` as data location. It is gas efficient to have `calldata` as memory location for reference type variables if the function's visibility is `external`

Recommendation:

We advise that the parameters with `memory` data location have their data location changed to `calldata`

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

ERC-10: Possibility of ether being trapped in the contract

Type	Severity	Location
Logical Issue	Major	ERC20Forwarder.sol L181, L211, L250, L283, L317, L352, L379, L408

Description:

The functions on the aforementioned lines are marked as `payable` yet they do not deal with ether amount sent along with the function call. The contract does not have a functionality to withdraw these ether inside the implementation contract and if the ethers are sent through the proxy contract then a new implementation would need to be deployed to withdraw the trapped ether.

Recommendation:

We advise to remove `payable` modifier from the signatures of the functions on the aforementioned lines to restrict the sending of ether with the function call.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERC-11: Ineffectual declaration of utilization of `SafeMath` library for `uint128`

Type	Severity	Location
Volatile Code	Informational	ERC20Forwarder.sol L29

Description:

The declaration `using SafeMath for uint128;` on the aforementioned line is ineffectual as there is no instance of utilization of it in the contract with the variables of type `uint128`. Additionally, the `SafeMath` library's function are intended to be called with `uint256` variables and using it for variables of type `uint128` will not protect against overflow/underflow of values.

Recommendation:

We advise to remove the statement `using SafeMath for uint128;` from the aforementioned line.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERR-01: The names of file and contract are different

Type	Severity	Location
Language Specific	Informational	ERC20ForwardRequestCompatible.sol L15

Description:

The contract `ERC20ForwardRequestTypes` is placed in a file with a different name.

Recommendation:

We advise to rename the contract file to `ERC20ForwardRequestTypes` to comply with the convention of naming Solidity files.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERF-01: Missing reason string

Type	Severity	Location
Coding Style	Informational	ERC20ForwarderProxy.sol L271

Description:

The `require` statement on the aforementioned line is missing reason string.

Recommendation:

We advise to add the reason string for the `require` message on the aforementioned line which aids readability and debuigging of code.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERF-02: Lack of verification for the passed constructor argument

Type	Severity	Location
Logical Issue	Medium	ERC20ForwarderProxy.sol L216

Description:

The constructor argument of `_admin` is not validated against zero value and it can result unwanted state of the contract where the admin functionality is no longer executable.

Recommendation:

We advise to perform the check against zero value for `_admin` parameter.

```
require(
   _admin != address(0),
   "_admin cannot be zero"
);
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERF-03: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	ERC20ForwarderProxy.sol L266

Description:

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation:

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERF-04: Order of contracts inheritance does not allow storage upgradeability of `ERC20ForwarderStorage`

Type	Severity	Location
Logical Issue	Minor	ERC20ForwarderProxy.sol L309

Description:

The order of contracts inheritance does not allow storage upgradeability if new storage variables are needed to be introduced in `ERC20ForwarderStorage` because any new storage variable introduced in the `ERC20ForwarderStorage` will overwrite the storage slot currently referenced by the variable `_owner`.

Recommendation:

We advise to to place `ERC20ForwarderStorage` after the `Ownable` contract in the inheritance chain so the storage of `ERC20Forwarder` contract could be updated or expanded by modifying `ERC20ForwarderStorage` contract.

```
contract ERC20ForwarderProxy is AdminUpgradeabilityProxy, Ownable,
ERC20ForwarderStorage {}
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERF-05: Lack of verification for the constructor argument

Type	Severity	Location
Logical Issue	Medium	ERC20ForwarderProxy.sol L10

Description:

The `_owner` constructor parameter is not validated against zero value and it can result in unwanted state of the contract where the owner specific functionality is no longer executable.

Recommendation:

We advise to perform the check against zero value for the parameter `_owner`.

```
    require(
       _owner != address(0),
       "_owner cannot be zero"
    );
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

ERS-01: Type `address payable` can be changed to `address`

Type	Severity	Location
Language Specific	Informational	ERC20ForwarderStorage.sol L23

Description:

The aforementioned line declares the state variable `forwarder` of type `address payable` yet no ether transfer is performed on the variable.

Recommendation:

We advise to change the type of the aforementioned variable from `address payable` to `address`.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

OAR-01: mappings data can be packed in a struct

Type	Severity	Location
Gas Optimization	Informational	OracleAggregator.sol L11-L14

Description:

The mappings on the aforementioned lines have key of type `address` representing a token. These mappings can be combined into a single mapping having `address` as key type and the value type can be a struct having properties from both of the mappings. This will reduce the lookup gas cost for these mappings.

Recommendation:

We advise to replace the aforementioned mappings with a single mapping by utilizing a struct for the value types across all the aforementioned mappings.

```
struct TokenInfo {
   uint8 decimals;
   bool dataSigned;
   address callAddress;
   bytes callData;
}
```

```
mapping(address => TokenInfo) internal tokensInfo;
```

The struct is laid out in a way such that the decimals, dataSigned and callAddress are packed into a single storage slot.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

OAR-02: Missing state variable's visibility

Type	Severity	Location
Language Specific	Informational	OracleAggregator.sol L11-L14

Description:

The state variables on the aforementioned lines does not have their visibilities specified.

Recommendation:

We advise to specify the visibilities of the state variables on the aforementioned lines as `private` or `internal`.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The state variables were combined as a single mapping which had its visiblity specified as `internal`.

OAR-03: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	OracleAggregator.sol L22

Description:

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation:

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

EMR-01: Lack of verification of the constructor argument

Type	Severity	Location
Logical Issue	Medium	ExecutorManager.sol L23

Description:

The constructor parameter `owner` on the aforementioned line is not validated against zero value and it can result in an unwanted state of the contract where the owner specific functionalities are no longer executable.

Recommendation:

We advise to perform the check against zero value for the parameter `owner`.

```
require(
   owner != address(0),
   "owner cannot be zero"
);
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

EMR-02: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	ExecutorManager.sol L38, L53

Description:

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation:

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-01: Unspecified state variable visibility

Type	Severity	Location
Language Specific	Informational	LiquidityPoolManager.sol L19

Description:

The state variable `executorManager` does not have its visibility specified.

Recommendation:

We advise to specify the visibility of the state variable on the aforementioned line to `private` or `internal`.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The state variable's visiblity is specified as `private`.

LPM-02: mappings data can be packed in a struct

Type	Severity	Location
Gas Optimization	Informational	LiquidityPoolManager.sol L21-L27

Description:

The mappings on the aforementioned lines have key of type `address` representing a token. These mappings can be combined into a single mapping having `address` as key type and the value type can be a struct having properties from both of the mappings. This will reduce the lookup gas cost for these mappings.

Recommendation:

We advise to replace the aforementioned mappings with a single mapping by utilizing a struct for the value types across all the aforementioned mappings.

```
struct TokenInfo {
   uint256 transferOverhead;
   bool supportedToken;
   uint256 minCap;
   uint256 maxCap;
   uint256 liquidity;
   mapping(address => uint256) liquidityProvider;
}
```

```
mapping(address => TokenInfo) public tokensInfo;
```

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-03: Lack of verification for the constructor parameters

Type	Severity	Location
Logical Issue	Medium	LiquidityPoolManager.sol L53

Description:

The constructor parameters of `owner`, `_trustedForwarder` and `_adminFee` are not validated against zero value and if the `owner` is set to zero address then it will result in an unwanted state of the contract where the owner specific functionalities are no longer executable.

Recommendation:

We advise to perform the check against zero value for the constructor parameters on the aforementioned lines.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-04: Mutability Specifiers Missing

Type	Severity	Location
Gas Optimization	Informational	LiquidityPoolManager.sol L17

Description:

The linked variables are assigned to only once, either during their contract-level declaration or during the `constructor` 's execution.

Recommendation:

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables. Please note that the `immutable` keyword only works in Solidity versions `v0.6.5` and up.

Alleviation:

A function was added to change the aforementioned state variable rendering this exhibit ineffectual.

LPM-05: `LiquidityAdded` event is not emitted

Type	Severity	Location
Volatile Code	Minor	LiquidityPoolManager.sol L106

Description:

The function `addNativeLiquidity` on the aforementioned line does not emit the event `LiquidityAdded`.

Recommendation:

We advise to emit the event `LiquidityAdded` inside body of the function on the aforementioned line.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-06: Use of `send` to transfer ether

Type	Severity	Location
Volatile Code	Minor	LiquidityPoolManager.sol L117

Description:

The aforementioned line performs sending of ether by calling `send` method on the address type. It only forwards 2300 gas which might not be enough if the receving address is a contract and consumes more gas than 2300.

Recommendation:

We advise to utilize `sendValue` function of the library `Address` and mark the function `nonRenentrant` to prevent reentrancy of the contract.

Address.sol

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

LPM-07: `LiquidityRemoved` event is not emitted

Type	Severity	Location
Volatile Code	Minor	LiquidityPoolManager.sol L131

Description:

The function `removeTokenLiquidity` on the aforementioned line does not emit the event `LiquidityRemoved`.

Recommendation:

We advise to emit the event `LiquidityRemoved` inside the body of the function on the aforementioned line.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-08: Usage of `msg.sender` instead of `_msgSender()`

Type	Severity	Location
Logical Issue	Major	LiquidityPoolManager.sol L155, L219, L221, L118

Description:

The aforementioned lines utilize `msg.sender` to retrieve the sender of the transaction. As the `LiquidityPoolManager` contract is a recipient contract capable of receiving offline signed transactions, it should not directly utilize `msg.sender` and instead make use of `_msgSender()` function to retrieve the message signer. Usage of `msg.sender` can result in unwanted behaviour such as on `L219` where the funds are transferred to the `Forwarder` instead of the actual message signer and the event on `L118` , `L155` and `L221` expect to receive the message signer address instead of the forwarder's address.

Recommendation:

We advise to utilize the function `_msgSender()` instead of directly using `msg.sender` on the aforementioned lines.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

LPM-09: Redundant code

Type	Severity	Location
Gas Optimization	Informational	LiquidityPoolManager.sol L163-L172

Description:

The code on the aforementioned lines is redundant and can be replaced by a call to the function `checkHashStatus` to determine if the hash is already processed.

Recommendation:

We advise to utilize the function `checkHashStatus` to determine the status of the hash. This will reduce the bytecode footrpint of the contract resulting reduce gas cost upon deployment.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-10: Incorrect comparison

Type	Severity	Location
Logical Issue	Minor	LiquidityPoolManager.sol L183, L186

Description:

The greater-than comparisons on the aforementioned lines does not allow transfer of whole balance of the contract as the contract's balance always has to be greater than the amount being transferred.

Recommendation:

We advise to change the comparison from greater-than to greater-than-or-equal such that all the balance from the contract could be moved.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`.

LPM-11: Use of `send` to transfer ether

Type	Severity	Location
Volatile Code	Minor	LiquidityPoolManager.sol L184

Description:

The aforementioned line performs sending of ether by calling `send` method on the address type. It only forwards 2300 gas which might not be enough if the receving address is a contract and consumes more gas than 2300.

Recommendation:

We recommend to utilize `sendValue` function of `Address` library to perform the sending of ether.

Address.sol

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1` .

LPM-12: Contract receives plain ether with no side-effects

Type	Severity	Location
Volatile Code	Medium	LiquidityPoolManager.sol L224

Description:

The function `receive` on the aforementioned line allows the contract to receive ether but such ether transfer has no side-effect unlike the ether received through `depositNative` function where the `Deposit` event is emitted with the intent to receive the amount across the side-chain.

Recommendation:

We advise to remove the `receive` function on the aforementioned line to avoid loss of user's funds by accidental sending.

Alleviation:

Alleviations were applied as of commit hash `f785038d30530ade41dbba93f93af78a980329c1`. The `recieve` function is removed from the contract.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.