Automated Arbitrage Bot

With the flash-loan arbitrage smart contract finished, our last task is to automate the process of finding and back-running other users' trades.

Since we already have a full bot that you can look at for a complete example, we'll just lay out the core principles and methods here, without walking through a whole bot step by step. This guide aims to give you everything you need to write your own edition ofsimple-blind-arbitrage .

For a more in-depth guide on writing a bot from scratch, check out theMEV-Share Limit Order Bot tutorial .

Client Libraries

To write your own bot, you'll need a way to interact with Flashbots APIs. Client libraries implement all the functionality of the APIs in their native programming language. If your preferred language doesn't have a MEV-Share client library already, you can interact with the APIs directly, effectively implementing your own client (let us know if you do!). Information about Flashbots APIs can be found in theRPC docs andEvent Stream docs .

Client libraries have been developed for the following programming languages (more to come soon):

typescript/javascript mev-share-client-ts (npm) rust mev-share-rs (crates.io)

Finding Pending Transactions

To reiterate our goal, we need to find pending transactions from other Ethereum users and backrun them (send our transaction immediately after the user's transaction). If the user's trade moves the price enough, we'll arbitrage the trading pair between two exchanges for a profit.

As MEV-Share receives new transactions, it broadcasts them to searchers via theEvent Stream . Each transaction shares varying amounts of data viahints , but by default, transactions that trade on Uniswap, Balancer, or Curve will expose the trading pair's contract address in the log topics.

Here's an example of an event generated by a user who's (most likely) using MEV-Share's default privacy settings:

Example Event

{ "hash": "0x0c459dce812747c643f06c82eeca2a2f584b4e30af79b2b546fd015e4aac4541", "logs": [{ "address": "0xca25091555d36ac0be8119ad967898ac30223b41", "topics": ["0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822", "0x0000000000000000000000000000000000000000000000000000000000000000", "0x0000000000000000000000000000000000000000000000000000000000000000"], "data": "0x" }, { "address": "0xca25091555d36ac0be8119ad967898ac30223b41", "topics": ["0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822", "0x0000000000000000000000000000000000000000000000000000000000000000", "0x0000000000000000000000000000000000000000000000000000000000000000"], "data": "0x" }], "txs": null } If welook up the address specified in the log topics, we'll see that it's a trading pair contract for QPEPE/WETH on Uniswap V2. This tells us that the user is buying/selling QPEPE on Uniswap, which means that we may be able to arbitrage it with Sushiswap or another Uni-V2 derivative.

To detect events yourself, listen to the SSE Event Stream athttps://mev-share.flashbots.net .

Events are comprised of either a singletransaction or abundle (multiple transactions):

- In abundle
- event, thehash
- field is thebundle hash
- and thetxs
- field will be populated with transaction-related data for each tx in the bundle.
- In a single-transaction
- event,hash
- represents thetransaction hash
- and thetxs
- field isnull

- .

- TS/JS

- Rust
- curl

const authSigner =

new

Wallet (Env . authKey) . connect (provider) const mevshare = MevShareClient . useEthereumMainnet (authSigner)

mevshare . on ("transaction" ,

async

(pendingTx : IPendingTransaction)

=>

{ // handle tx event }) In this library, transactions are natively differentiated from bundles, so if you want to detect bundles too, just add another handler usingon("bundle", ...) :

mevshare . on ("bundle" ,

async

(pendingBundle : IPendingBundle)

=>

{ // handle bundle event }) Check out theimplementation for a closer look at how events are defined. let mainnet_sse =

"https://mev-share.flashbots.net" ; let client =

EventClient :: default ( ) ; let

mut stream = client . events (mainnet_sse) . await . unwrap ( ) ; while

let

Some (event)

= stream . next ( ) . await

{ // handle event } In this library, an event whosetxs field is null is encoded as an empty array, so in practice, we need to check ifevent.txs is empty to see if we have a transaction or a bundle.

while

let

Some (event)

= stream . next ( ) . await

{ if event . txs . len ( )

==

0

{ // handle single tx }

else

{ // handle bundle } } curl https://mev-share.flashbots.net The Event Stream is streamed via SSE over a simple HTTP GET request, which may seem confusing. It's not terribly important for the purpose of building bots, but if you're curious, SSE wasadded to HTML in HTML5.

To read more about how the SSE stream works, see theEvent Stream docs .

Filtering Relevant Transactions

MEV-Share useshints to selectively share information about a transaction. Based on the hint preferences specifies by the user when connecting, the transactions (or bundles) they send will trigger events containing information about their transaction which is filtered according to their hint preferences.

In this guide, we're only concerned with the fields inlogs :address andtopics . Other fields not covered in this guide are detailed in theEvent Scheme docs .

Snippet from an example event:

{ ..., "logs": [{ "address": "0xca25091555d36ac0be8119ad967898ac30223b41", "topics": ["0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822", "0x0000000000000000000000000000000000000000000000000000000000000000", "0x0000000000000000000000000000000000000000000000000000000000000000"], "data": "0x" }, ...], } address tells us which contract address the user is interacting with. Looking back at theexample event , you'll notice that theaddress is thetrading pair , and not a router contract. This is likely because the user is using the default hints, which expose the trading pair instead of whatever router they might be interacting with. It's also possible that the user is another searcher, and they're trading on the pair directly. At any rate, it makes no difference to us.

Given the default hint preferences, MEV-Share also exposesonly swap-related function signatures (if present) in thetopics . Thesimple-blind-arbitrage bot uses theUniswap V2 Swap event signature to find pending Uniswap V2 trades.

To find event signatures yourself, you need to take the keccak256 hash of the event signature. A nice tool to use for this iscast from Foundry :

get the hash of the UniV2 Swap event

cast sig-event "event Swap(address indexed sender, uint amount0In, uint amount1In, uint amount0Out, uint amount1Out, address indexed to);" 0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822 To filter our results so that we only deal with Swap events, we can simply check thelogs field. If it isn't empty, each entry should containtopics . If our event signature hash is in those topics, then we know we're looking at a swap that we can backrun. For events generated by Protect users, the relevant signature hash is the first topic in the array, so we checklog.topics[0] .

Once we find a pending swap on one exchange, we need to find another exchange to arbitrage with. For example, if we detect a Uniswap V2 event, then we should try to arbitrage with Sushiswap. To get the trading pair on the other exchange, we need to use that exchange'sFactory contract, which maps pairs of token addresses toPair contract addresses.

- TS/JS
- Rust

import

{ Contract }

from

"ethers" import uniV2FactoryABI from

"./abi/uniswapV2Factory.json"

const swapTopic =

"0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822"

// instantiate factory contracts const uniV2FactoryAddress =

"0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f" const uniV2Factory =

new

Contract (uniV2FactoryAddress , uniV2FactoryABI) const sushiFactoryAddress =

"0xC0AEe478e3658e2610c5F7A4A2E1777cE9e4f2Ac" const sushiFactory =

new

Contract (sushiFactoryAddress , uniV2FactoryABI)

mevshare . on ("transaction" ,

async

(tx : IPendingTransaction)

=>

{ for

(const log of tx . logs)

{ // skip if it isn't a swap event if

(log . topics [0]

!== swapTopic)

{ continue }

// data needed to find arb pair on another exchange const pair =

new

ethers . Contract (log . address , pairABI) const token0 =

await pair . token0 ( ) const token1 =

await pair . token1 ( ) // primitive differentiator between uniswap & sushiswap. ideally you'd use an enum to support >2 exchanges. const isUniswap = pair . factory === uniV2FactoryAddress

const altFactory = isUniswap ? sushiFactory : uniV2Factory const altPair =

await altFactory . getPair (token0 , token1) if

(altPair ===

"0x0000000000000000000000000000000000000000")

{ console . error ("pair not found on alternative exchange") continue }

// placeholder: send backrun bundle await

tryBackrun (pair . address , altPair) } }) Note: querying the blockchain can cost precious time -- a better design would be to store arb-ready pairs somewhere fast, like in memory or a fast DB, and only query the blockchain when a stored pair alternative can't be found. let swap_topic =

"0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822" . parse :: < H256

( ) . unwrap ( ) ; while

let

Some (event)

= stream . next ( ) . await

{ dbg! (& event) ; // handle tx or bundle events containing swap logs let event = event . unwrap ( ) ; for log in event . logs { // skip if it isn't a swap event if log . topics [0]

!= swap_topic { continue ; }

// get data needed to find arb pair // TODO

// placeholder: send backrun bundle tryBackrun (pair . address , altPair . address) . await ? ; } } Note that the Sushiswap factory uses the same ABI as the Uniswap contract. Sushiswap is just a Uniswap V2 clone, so we know that the code is the same, which means the ABIs are also the same.

blind backruns Note that we potentially send a bundle for every log in the array. This is because some trades affect many trading pairs. Because we don't know which one was affected, we just attempt to backrun them all. The bundles we'll send for a given opportunity will use the same nonce for every backrun tx, so only the most profitable one will land.

Sending Backrun Bundles

So far we've seen how to find the events we want to backrun, and we have a placeholder (tryBackrun) in place to send our backrun bundles. Once we implementtryBackrun , we'll have everything we need to send backrun bundles.

A backrun bundle is an array consisting of two or more transactions: the user's transaction, and the following "backrun transactions." The backrun transaction we'll send is a call tomakeFlashLoan on our arbitrage contract. If you recall from the implementation of theflash loan arbitrage contract , thereceiveFlashLoan function calls_executeArbitrage before paying back the loan. So all we have to do is get a flash loan, and then our contract will try to execute an arbitrage with the tokens it receives. If it's profitable, we pay the loan back, pay the builder, and keep the rest. If not, it reverts and is discarded by the builder.

This is the function we need to call:

function

makeFlashLoan (IERC20 [ ]

memory tokens , uint256 [ ]

memory amounts , bytes

memory userData) To callmakeFlashLoan , we specify thetokens we want to borrow, the respectiveamounts of tokens to borrow, and the ABI-encodeduserData which will contain the arguments to the_executeArbitrage function.

- TS/JS
- Rust

abi/blindBackrunFlashLoan.json { "abi": [{ "inputs": [{ "internalType": "address", "name": "wethAddress", "type": "address" }], "stateMutability": "nonpayable", "type": "constructor" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "address", "name": "previousOwner", "type": "address" }, { "indexed": true, "internalType": "address", "name": "newOwner", "type": "address" }], "name": "OwnershipTransferred", "type": "event" }, { "inputs": [], "name": "WETH_ADDRESS", "outputs": [{ "internalType": "address", "name": "", "type": "address" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "address payable", "name": "_to", "type": "address" }, { "internalType": "uint256", "name": "_value", "type": "uint256" }, { "internalType": "bytes", "name": "_data", "type": "bytes" }], "name": "call", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "address", "name": "firstPairAddress", "type": "address" }, { "internalType": "address", "name": "secondPairAddress", "type": "address" }, { "internalType": "uint256", "name": "percentageToPayToCoinbase", "type": "uint256" }, { "internalType": "address", "name": "recipient", "type": "address" }], "name": "executeArbitrage", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple" }, { "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "secondPairData", "type": "tuple" }], "name": "getAmountIn", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple" }, { "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "secondPairData", "type": "tuple" }], "name": "getDenominator", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple" }, { "components": [{ "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" }], "internalType": "struct IPairReserves.PairReserves", "name": "secondPairData", "type": "tuple" }], "name": "getNumerator", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "contract IERC20[]", "name": "tokens", "type": "address[]" }, { "internalType": "uint256[]", "name": "amounts", "type": "uint256[]" }, { "internalType": "bytes", "name": "userData", "type": "bytes" }], "name": "makeFlashLoan", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "owner", "outputs": [{ "internalType": "address", "name": "", "type": "address" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "contract IERC20[]", "name": "tokens", "type": "address[]" }, { "internalType": "uint256[]", "name": "amounts", "type": "uint256[]" }, { "internalType": "uint256[]", "name": "feeAmounts", "type": "uint256[]" }, { "internalType": "bytes", "name": "userData", "type": "bytes" }], "name": "receiveFlashLoan", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "renounceOwnership", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "address", "name": "newOwner", "type": "address" }], "name": "transferOwnership", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "withdrawETHToOwner", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "withdrawWETHToOwner", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "stateMutability": "payable", "type": "receive" }], "bytecode": { "object":
"0x60a06040526103e56001553480156100165760080fd5b50604051611a56380380611a5683398101604081905261003591610 0a1565b8061003f33610051565b6001600160a01b0316608052506100c
"sourceMap": "1103:1241:26:-:0;;;941:3:23;918:26;;1267:63:26;;;;;;;;;;;;;;;;;;;;;;;;;;;:::i;:::-;1314:12;936:32:0;719:10:4;936:18:0;:32::i;:::-;-1:-1:-1;;;;;1039:27:23;;;-1:-1:-1;1103:1241:26;;2426:187:0;2499:16;2518:6;;-1:-1:-1;;;;;2534:17:0;;;-1:-1:-1;;;;;;2534:17:0;;;;;;2566:40;;2518:6;;;;;;;2566:40;;2499:16;2566:40;2489:124;2426:187;:::o;14:290:28:-;84:6;137:2;125:9;116:7;112:23;108:32;105:52;;;153:1;150;143:12;105:52;179:16;;-1:-1:-1;;;;;224:31:28;;214:42;;204:70;;270:1;267;260:12;204:70;293:5;14:290;-1:-1:-1;;;14:290:28:o;:::-;1103:1241:26;;;", "linkReferences": {} }, "deployedBytecode": { "object":
"0x608060405260043610610 0c45760003560e01c806377432b8c1161007f578063ea3e506c11610059578063ea3e506c14610219578063eb6b21161461022e578063f04f27071461024e578063f2fde38b1
"sourceMap": "1103:1241:26:-:0;;;4524:1133:23;;;;;;;;;-1:-1:-1;4524:1133:23;;;;;:::i;:::-;;:::i;:::-;;;1750:25:28;;;1738:2;1723:18;4524:1133:23;;;;;;;1424:2330;;;;;;;;;-1:-1:-1;1424:2330:23;;;;;:::i;:::-;;:::i;:::-;951:37;;;;;;;;;;;;;;;;-1:-1:-1;;;;;2689:32:28;;;2671:51;;2659:2;2644:18;951:37:23;2525:203:28;5663:687:23;;;;;;;;;-1:-1:-1;5663:687:23;;;;;:::i;:::-;;:::i;9330:236::-;;;;;;;;;-1:-1:-1;9330:236:23;;;;;:::i;:::-;;:::i;1824:101:0:-;;;;;;;;;;;;;;:::i;8794:152:23:-;;;;;;;;;;;;;;:::i;1201:85:0:-;;;;;;;;;-1:-1:-1;1247:7:0;1273:6;-1:-1:-1;;;;;1273:6:0;1201:85;;1336:210:26;;;;;;;;;-1:-1:-1;1336:210:26;;;;;:::i;:::-;;:::i;8459:190:23:-;;;;;;;;;;;;;;:::i;4097:421::-;;;;;;;;;-1:-1:-1;4097:421:23;;;;;:::i;:::-;;:::i;1552:790:26:-;;;;;;;;;-1:-1:-1;1552:790:26;;;;;:::i;:::-;;:::i;2074:198:0:-;;;;;;;;;-1:-1:-1;2074:198:0;;;;;:::i;:::-;;:::i;4524:1133:23:-;4711:24;;;;4688:7;;4711:32;;4739:4;4711:32;4707:944;;4759:12;4774:229;4980:13;:22;;;4774:184;4934:14;:23;;;4774:138;4888:14;:23;;;4774:92;4843:13;:22;;;4774:47;4808:12;;4774;;:33;;:47;;;;:::i;:::-;:68;;:92::i;:138::-;:159;;:184::i;:229::-;4759:244;;5018:17;5038:115;5130:13;:22;;;5038:70;5084:14;:23;;;5039:22;5057:3;5039:13;5044:7;5039:4;:13::i;:::-;:17;;:22::i;5038:115::-;5018:135;-1:-1:-1;5168:16:23;;-1:-1:-1;;5168:16:23;4707:944;5215:12;5230:229;5436:13;:22;;;5230:184;5390:14;:23;;;5230:138;5344:14;:23;;;5230:92;5299:13;:22;;;5230:47;5264:12;;5230;;:33;;:47;;;;;:::i;:229::-;5215:244;;5474:17;5494:115;5586:13;:22;;;5494:70;5540:14;:23;;;5495:22;5513:3;5495:13;5500:7;5495:4;:13::i;4707:944::-;4524:1133;;;;;:::o;1424:2330::-;1633:45;;-1:-1:-1;;;1633:45:23;;1672:4;1633:45;;;2671:51:28;1609:21:23;;1640:12;-1:-1:-1;;;;;1633:30:23;;;;2644:18:28;;1633:45:23;;;:::i;:::-;1609:69;-1:-1:-1;1730:16:23;1800:17;1688:24;1879:44;1730:16;1879:11;:44::i;:::-;1829:94;;1933:48;1984:45;2009:10;1984:11;:45::i;:::-;1933:96;;2040:16;2059:42;2071:13;2086:14;2059:11;:42::i;:::-;2111:57;;-1:-1:-1;;;2111:57:23;;-1:-1:-1;;;;;7933:32:28;;;2111:57:23;;;7915:51:28;7982:18;;;7975:34;;;2040:61:23;;-1:-1:-1;2118:12:23;2111:29;;;;;;7888:18:28;;2111:57:23;;;:::i;:::-;;2179:26;2215:22;2251:13;:24;;;:32;;2279:4;2251:32;;;2247:1034;;2320:132;2350:8;2376:13;:22;;;2416:13;:22;;;2320:12;:132::i;:::-;2299:153;;2483:144;2513:18;2549:14;:23;;;2590:14;:23;;;2483:12;:144::i;:::-;2466:161;;2642:9;-1:-1:-1;;;;;2642:14:23;;2657:1;2660:18;2680:17;2642:60;;;;;;;;;;;;;;;;;:::i;:::-;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-1:-1:-1;;2716:53:23;;-1:-1:-1;;;2716:53:23;;-1:-1:-1;;;;;2716:15:23;;;-1:-1:-1;2716:15:23;;-1:-1:-1;2716:53:23;;2732:14;;2748:1;;2759:4;;2716:53;;;;:::i;:::-;;2247:1034;;;2821:132;2851:8;2877:13;:22;;;2917:13;:22;;;2821:12;:132::i;:::-;2800:153;;2984:144;3014:18;3050:14;:23;;;3091:14;:23;;;2984:12;:144::i;:::-;2967:161;;3143:9;-1:-1:-1;;;;;3143:14:23;;3158:18;3178:1;3181:17;3143:60;;;;;;;;;;;;;;;;;:::i;:::-;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-1:-1:-1;;3217:53:23;;-1:-1:-1;;;3217:53:23;;-1:-1:-1;;;;;3217:15:23;;;-1:-1:-1;3217:15:23;;-1:-1:-1;3217:53:23;;3233:1;;3236:14;;3260:4;;3217:53;;;;:::i;:::-;;2247:1034;3314:45;;-1:-1:-1;;;3314:45:23;;3353:4;3314:45;;;2671:51:28;3291:20:23;;3321:12;-1:-1:-1;;;;;3314:30:23;;;;2644:18:28;;3314:45:23;;;:::i;:::-;3291:68;;3392:13;3377:12;:28;3369:57;;;;-1:-1:-1;;;3369:57:23;;9548:2:28;3369:57:23;;;9530:21:28;9587:2;9567:18;;;9560:30;-1:-1:-1;;;9606:18:28;;;9599:46;9662:18;;3369:57:23;;;;;;;;;3436:11;3450:31;:12;3467:13;3450:16;:31::i;:::-;3436:45;;3491:21;3515:46;3557:3;3515:37;3526:25;3515:6;:10;;:37;;;;:::i;:46::-;3571;;-1:-1:-1;;;3571:46:23;;;;;1750:25:28;;;3491:70:23;;-1:-1:-1;3577:12:23;-1:-1:-

1;;;;;3571:28:23;;;;1723:18:28;;3571:46:23;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-1:-1:-1;;3627:41:23;;:14;;-1:-1:-1;3627:41:23;;;;;-1:-1:-1;3651:16:23;;3627:41;;;;3651:16;3627:14;:41;;;;;;;;;;;;;;;;;;;-1:-1:-
1;;;;;;3684:12:23;3678:28;;3707:9;3718:28;:6;3729:16;3718:10;:28::i;:::-;3678:69;;-1:-1:-1;;;;;;3678:69:23;;;;;;;-1:-1:-
1;;;;;7933:32:28;;;3678:69:23;;;7915:51:28;7982:18;;;7975:34;7888:18;;3678:69:23;;::i;:::-;;1599:2155;;;;;;;;;;1424:2330;;;;:::o;5663:687::-
;5852:24;;;;5829:7;;5852:32;;5880:4;5852:32;5848:496;;5900:19;5922:147;6010:58;6045:13;:22;;;6010:30;6027:12;;6010;;:16;;:30;;;;;:::i;:58::-
;5940:51;5986:4;5940:41;5957:14;:23;;;5940:12;;:16;;:41;;;;;:::i;:51::-;5922:87;;:147::i;:::-;5900:169;-1:-1:-1;6083:18:23;;-1:-1:-
1;6083:18:23;5848:496;6277:22;;6259:12;;6132:19;;6154:147;;6242:58;;6277:22;6242:30;;6259:12;6242:16;:30::i;:58::-;6189:23;;6172:12;;:51;;6218:4;;6172:41;;:16;:41::i;9330:236::-
;1094:13:0;:11;:13::i;:::-;9461:12:23::1;9479:3;-1:-1:-1::0;;;;;9479:8:23::1;9495:6;9503:5;9479:30;;;;;;;:::i;:::-;;;9460:49;;;9527:7;9519:40;;;:::0;-1:-1:-
1;;;9519:40:23;;10440:2:28;9519:40:23::1;;:::0;;:::1;10422:21:28::0;10479:2;10459:18;;;10452:30;-1:-1:-
1;;;10498:18:28;;;10491:50;10558:18;;9519:40:23::1;10238:344:28::0;9519:40:23::1;9450:116;9330:236:::0;;;:::o;1824:101:0:-;1094:13;:11;:13::i;:::-;1888:30:::1;1915:1;1888:18;:30::i;:::-
;1824:101::o:0;8794:152:23:-;1094:13:0;:11;:13::i;:::-
;8902:37:23::1;;:::0;8871:21:::1;;:::0;8910:10:::1;;:::0;8902:37;;:::1;;;;:::0;8871:21;;8853:15:::1;8902:37:::0;8853:15;8902:37;8871:21;8910:10;8902:37;;:::1;;;;;;;;;;;;;;:::0;;:::1;;;;;;8843:103;8794:152::o:0;1336
;1094:13:0;:11;:13::i;:::-;1491:48:26::1;;:::0;-1:-1:-1;;;1491:48:26;;1217:42:::1;;:::0;1491:15:::1;;:::0;:48:::1;;:::0;1507:4:::1;;:::0;1513:6;;1521:7;;1530:8;;1491:48:::1;;;;:::i;:::-
;;;;;;;;;;;;;;;;;;;:::0;;:::1;;;;;;;;;;;;;;:::0;;:::1;;;;;;;;;1336:210:::0;;;;:::o;8459:190:23:-;1094:13:0;:11;:13::i;:::-;8537:45:23::1;;:::0;-1:-1:-
1;;;8537:45:23;;8576:4:::1;8537:45;;:::0;;:::1;2671:51:28::0;8519:15:23::1;;:::0;8544:12:::1;;-1:-1:-1::0;;;;;8537:30:23::1;;:::0;;:::1;;:::0;2644:18:28;;8537:45:23::1;;;;;;;;;;;;;;;;;;;;:::0;;:::1;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;::i;:::-
;8592:50;;:::0;-1:-1:-1;;;8592:50:23;;8622:10:::1;8592:50;;:::0;;:::1;7915:51:28::0;7982:18;;;7975:34;;;8519:63:23;;-1:-1:-1;8599:12:23::1;;-1:-1:-
1::0;;;;;8592:29:23::1;;:::0;;:::1;;:::0;7888:18:28;;8592:50:23::1;;;;;;;;;;;;;;;;;;;;;:::0;;:::1;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;::i;4097:421::-:0;4260:7;4279:17;4299:43;4312:13;4327:14;4299:12;:43::i;:::-
;4279:63;;4352:19;4374:45;4389:13;4404:14;4374;:45::i;:::-;4352:67;-1:-1:-1;4430:16:23;4449:36;4352:67;4449:19;:9;4463:4;4449:13;:19::i;:36::-;4430:55;4097:421;-1:-1:-
1;;;;;;4097:421:23:o;1552:790:26:-;1758:10;1217:42;1758:28;1737:118;;;;-1:-1:-1;;;1737:118:26;;12517:2:28;1737:118:26;;;12499:21:28;12556:2;12536:18;;;12529:30;12595:34;12575:18;;;12568:62;-
1:-1:-1;;;12646:18:28;;;12639:41;12697:19;;1737:118:26;12315:407:28;1737:118:26;1880:24;1918:25;1957:33;2014:8;2003:49;;;;;;;;;;;;;:::i;:::-
;1866:186;;;;;;2063:153;2093:17;2124:16;2154:25;2201:4;2063:16;:153::i;:::-;2233:12;-1:-1:-1;;;;;2227:28:26;;1217:42;2311:10;2322:1;2311:13;;;;;;;;;:::i;:::-;;;;;;;2297:7;2305:1;2297:10;;;;;;;;;:::i;:::-
;;;;;;;:28;;;;;;:::i;:::-;2227:108;;-1:-1:-1;;;;;;2227:108:26;;;;;;-1:-1:-1;;;;;;7933:32:28;;;2227:108:26;;;7915:51:28;7982:18;;;7975:34;7888:18;;2227:108:26;;::i;:::-
;;1727:615;;;1552:790;;;;;:::o;2074:198:0:-;1094:13;:11;:13::i;:::-;-1:-1:-1;;;;;2162:22:0;;;:::1;2154:73;;;;;;:::0;;-1:-1:-
1;;;2154:73:0;;13790:2:28;2154:73:0::1;;:::0;;:::1;13772:21:28::0;13829:2;13809:18;;;13802:30;13868:34;13848:18;;;13841:62;-1:-1:-
1;;;13919:18:28;;;13912:36;13965:19;;2154:73:0::1;13588:402:28::0;2154:73:0::1;2237:28;2256:8;2237:18;:28::i;:::-;2074:198::0;::::o;3465:96:5:-;3523:7;3549:5;3553:1;3549;:5;:::i;:::-;3542:12;3465:96;-
1:-1:-1;;;3465:96:5:o;3850::::-;3908:7;3934:5;3938:1;3934;:5;:::i;7359:268:23:-;7406:7;7429:1;7434;7429:6;7425:20;;-1:-1:-1;7444:1:23;;7359:268;-1:-1:-1;7359:268:23:o;7425:20::-
;7455:9;7467:15;7480:1;7467:8;:1;7473;7467:5;:8::i;:15::-;7455:27;-1:-1:-1;7504:1:23;7515:88;7526:1;7522;:5;7515:88;;;-1:-1:-
1;7547:1:23;7566:26;7590:1;7567:17;7547:1;7568:8;:1;7547;7568:5;:8::i;7566:26::-;7562:30;;7515:88;;3122:96:5;3180:7;3206:5;3210:1;3206;:5;:::i;6621:552:23:-;6699:33;-1:-1:-
1;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;6699:33:23;6745:16;6763;6785:4;-1:-1:-1;;;;;6785:16:23;;:18;;;::i;:::-;6744:59;-1:-1:-1;;;;;6744:59:23;;;-1:-1:-
1;;;;;6744:59:23;;;6813:13;6837:15;6855:5;6837:23;;6891:12;-1:-1:-1;;;;;6874:29:23;:4;-1:-1:-1;;;;;6874:11:23;;:13;;::i;:::-;-1:-1:-
1;;;;;6874:29:23;;6870:202;;6927:32;6950:8;6927:18;:8;6940:4;6927:12;:18::i;:32::-;6919:40;;6986:4;6973:17;;6870:202;;;7029:32;7052:8;7029:18;:8;7042:4;7029:12;:18::i;:32::-
;7021:40;;6870:202;7101:65;;;;;;;;;;;;;;;;;;;;;;;;;;;;6621:552;-1:-1:-1;;6621:552:23:o;7920:393::-;8042:14;;8091:17;:8;8104:3;8091:12;:17::i;:::-;8068:40;-1:-1:-
1;8118:14:23;8135:31;8068:40;8155:10;8135:19;:31::i;:::-;8118:48;-1:-1:-1;8176:16:23;8195:40;8219:15;8195:19;:9;8209:4;8195:13;:19::i;:40::-;8176:59;-1:-1:-1;8257:23:23;8176:59;8257:9;:23;:::i;:::-
;8245:35;7920:393;-1:-1:-1;;;;;;7920:393:23:o;2755:96:5:-;2813:7;2839:5;2843:1;2839;:5;:::i;1359:130:0:-;1247:7;1273:6;-1:-1:-1;;;;;1273:6:0;719:10:4;1422:23:0;1414:68;;;;-1:-1:-
1;;;1414:68:0;;15629:2:28;1414:68:0;;;15611:21:28;;;15648:18;;;15641:30;15707:34;15687:18;;;15680:62;15759:18;;1414:68:0;15427:356:28;2426:187:0;2499:16;2518:6;;-1:-1:-1;;;;;2534:17:0;;;-1:-1:-
1;;;;;;2534:17:0;;;;;;2566:40;;2518:6;;;;;;;2566:40;;2499:16;2566:40;2489:124;2426:187;;;;:o;14:127:28:-
;75:10;70:3;66:20;63:1;56:31;106:4;103:1;96:15;130:4;127:1;120:15;146:275;217:2;211:9;282:2;263:13;;-1:-1:-1;;259:27:28;247:40;;;317:18;302:34;;;338:22;;;299:62;296:88;;;364:18;;::::i;:::-
;400:2;393:22;146:275;;-1:-1:-1;146:275:28:o;426:118::-
;512:5;505:13;498:21;491:5;488:32;478:60;;534:1;531;524:12;549:679;608:5;656:4;644:9;639:3;635:19;631:30;628:50;;;674:1;671;664:12;628:50;707:2;701:9;749:4;741:6;737:17;820:6;808:10;805:22;78
;871:10;867:2;860:22;;;900:6;891:15;;;943:9;930:23;922:6;915:39;1015:2;1004:9;1000:18;987:32;982:2;974:6;970:15;963:57;1081:2;1070:9;1066:18;1053:32;1048:2;1040:6;1036:15;1029:57;1138:2;1127
;1209:2;1197:15;;;;1190:32;549:679;;-1:-1:-1;;549:679:28:o;1233:366::-
;1363:6;1371;1424:3;1412:9;1403:7;1399:23;1395:33;1392:53;;;1441:1;1438;1431:12;1392:53;1464:50;1506:7;1495:9;1464:50;:::i;:::-;1454:60;;1533;1585:7;1579:3;1568:9;1564:19;1533:60;:::i;:::-
;1523:70;;1233:366;;;;;:::o;1786:131::-;-1:-1:-
1;;;;;1861:31:28;;1851:42;;1841:70;;1907:1;1904;1897:12;1922:598;2008:6;2016;2024;2032;2085:3;2073:9;2064:7;2060:23;2056:33;2053:53;;;2102:1;2099;2092:12;2053:53;2141:9;2128:23;2160:31;218
;2210:5;-1:-1:-1;2267:2:28;2252:18;;2239:32;2280:33;2239:32;2280:33;:::i;:::-;2332:7;-1:-1:-1;2386:2:28;2371:18;;2358:32;;-1:-1:-1;2442:2:28;2427:18;;2414:32;2455:33;2414:32;2455:33;:::i;:::-
;1922:598;;;;-1:-1:-1;1922:598:28;-1:-1:-1;;1922:598:28:o;2733:530::-
;2775:5;2828:3;2821:4;2813:6;2809:17;2805:27;2795:55;;2846:1;2843;2836:12;2795:55;2882:6;2869:20;2908:18;2904:2;2901:26;2898:52;;;2930:18;;::::i;:::-;2974:55;3017:2;2998:13;;-1:-1:-
1;;2994:27:28;3023:4;2990:38;2974:55;::::i;:::-
;3054:2;3045:7;3038:19;3100:3;3093:4;3088:2;3080:6;3076:15;3072:26;3069:35;3066:55;;;3117:1;3114;3107:12;3066:55;3182:2;3175:4;3167:6;3163:17;3156:4;3147:7;3143:18;3130:55;3230:1;3205:16
1:-1:-1;;;2733:530:28:o;3268:531::-;3362:6;3370;3378;3431:2;3419:9;3410:7;3406:23;3402:32;3399:52;;;3447:1;3444;3437:12;3399:52;3486:9;3473:23;3505:31;3530:5;3505:31;:::i;:::-;3555:5;-1:-1:-
1;3607:2:28;3592:18;;3579:32;;-1:-1:-1;3662:2:28;3647:18;;3634:32;3689:18;3678:30;;3675:50;;;3721:1;3718;3711:12;3675:50;3744:49;3785:7;3776:6;3765:9;3761:22;3744:49;:::i;:::-
;3734:59;;3268:531;;;;;:::o;3804:191::-;3872:4;3905:18;3897:6;3894:30;3891:56;;;3927:18;;::::i;:::-;-1:-1:-1;3972:1:28;3968:14;3984:4;3964:25;;3804:191::o;4000:753::-
;4062:5;4115:3;4108:4;4100:6;4096:17;4092:27;4082:55;;4133:1;4130;4123:12;4082:55;4169:6;4156:20;4195:4;4219:68;4235:51;4283:2;4235:51;:::i;:::-;4219:68;:::i;:::-
;4321:15;;;4407:1;4403:10;;;;4391:23;;4387:32;;;4352:12;;;;4431:15;;;4428:35;;;4459:1;4456;4449:12;4428:35;4495:2;4487:6;4483:15;4507:217;4523:6;4518:3;4515:15;4507:217;;;4603:3;4590:17;4620:3
;4664:18;;;4702:12;;;;4540;4507:217;;;-1:-1:-1;4742:5:28;4000:753;-1:-1:-1;;;;;;4000:753:28:o;4758:670::-
;4812:5;4865:3;4858:4;4850:6;4846:17;4842:27;4832:55;;4883:1;4880;4873:12;4832:55;4919:6;4906:20;4945:4;4969:68;4985:51;5033:2;4985:51;:::i;4969:68::-
;5071:15;;;5157:1;5153:10;;;;5141:23;;5137:32;;;5102:12;;;;5181:15;;;5178:35;;;5209:1;5206;5199:12;5178:35;5245:2;5237:6;5233:15;5257:142;5273:6;5268:3;5265:15;5257:142;;;5339:17;;5327:30;;537
;5807:79;;5939:2;5928:9;5924:18;5911:32;5895:48;;5968:2;5958:8;5955:16;5952:36;;;5984:1;5981;5974:12;5952:36;6007:63;6062:7;6051:8;6040:9;6036:24;6007:63;:::i;:::-
;5997:73;;6123:2;6112:9;6108:18;6095:32;6079:48;;6152:2;6142:8;6139:16;6136:36;;;6168:1;6165;6158:12;6136:36;;6191:51;6234:7;6223:8;6212:9;6208:24;6191:51;:::i;6253:1042::-
;6437:6;6445;6453;6461;6514:3;6502:9;6493:7;6489:23;6485:33;6482:53;;;6531:1;6528;6521:12;6482:53;6571:9;6558:23;6600:18;6641:2;6633:6;6630:14;6627:34;;;6657:1;6654;6647:12;6627:34;6680
;6670:79;;6802:2;6791:9;6787:18;6774:32;6758:48;;6831:2;6821:8;6818:16;6815:36;;;6847:1;6844;6837:12;6815:36;6870:63;6925:7;6914:8;6903:9;6899:24;6870:63;:::i;:::-
;6860:73;;6986:2;6975:9;6971:18;6958:32;6942:48;;7015:2;7005:8;7002:16;6999:36;;7031:1;7028;7021:12;6999:36;7054:63;7109:7;7098:8;7087:9;7083:24;7054:63;:::i;:::-
;7044:73;;7170:2;7159:9;7155:18;7142:32;7126:48;;7199:2;7189:8;7186:16;7183:36;;;7215:1;7212;7205:12;7183:36;;7238:51;7281:7;7270:8;7259:9;7255:24;7238:51;:::i;:::-
;7228:61;;;6253:1042;;;;;;;;:::o;7300:247::-;7359:6;7412:2;7400:9;7391:7;7387:23;7383:32;7380:52;;;7428:1;7425;7418:12;7380:52;7467:9;7454:23;7486:31;7511:5;7486:31;:::i;7552:184::-
;7622:6;7675:2;7663:9;7654:7;7650:23;7646:32;7643:52;;;7691:1;7688;7681:12;7643:52;-1:-1:-1;7714:16:28;;7552:184;-1:-1:-1;7552:184:28:o;8020:245::-
;8087:6;8140:2;8128:9;8119:7;8115:23;8111:32;8108:52;;;8156:1;8153;8146:12;8108:52;8188:9;8182:16;8207:28;8229:5;8207:28;:::i;8270:533::-;8545:25;;;8601:2;8586:18;;8579:34;;;;-1:-1:-
1;;;;;8649:32:28;8644:2;8629:18;;8622:60;8718:3;8713:2;8698:18;;8691:31;;;-1:-1:-1;;8738:19:28;;;8731:30;8669:3;8778:19;;8270:533::o;9691:250::-
;9776:1;9786:113;9800:6;9797:1;9794:13;9786:113;;;9876:11;;;9870:18;9857:11;;;9850:39;9822:2;9815:10;9786:113;;;-1:-1:-1;;9933:1:28;9915:16;;9908:27;9691:250::o;9946:287::-
;10075:3;10113:6;10107:13;10129:66;10188:6;10183:3;10176:4;10168:6;10164:17;10129:66;:::i;:::-;10211:16;;;;;9946:287;-1:-1:-1;;9946:287:28:o;10587:270::-
;10628:3;10666:5;10660:12;10693:6;10688:3;10681:19;10709:76;10778:6;10771:4;10766:3;10762:14;10755:4;10748:5;10744:16;10709:76;:::i;:::-;10839:2;10818:15;-1:-1:-
1;;10814:29:28;10805:39;;;;10846:4;10801:50;;10587:270;-1:-1:-1;;10587:270:28:o;10862:1448::-;-1:-1:-
1;;;;;11311:15:28;;;11293:34;;11242:3;11346:2;11364:18;;;11357:31;;;11437:13;11227:19;;;11459:22;;;11194:4;11539:15;;;11265:19;11346:2;11273:3;11497:19;;;11194:4;11582:178;11596:6;11593:1
1:-1:-1;;11796:19:28;;;11791:2;11776:18;;11769:47;11866:13;;11888:21;;;11927:12;;;;-1:-1:-1;;11866:13:28;;-1:-1:-
1;11964:15:28;;;11999:1;12009:189;12025:8;12020:3;12017:17;12009:189;;;12094:15;;12080:30;12132:14;;;;12171:17;;;;12053:1;12044:11;12009:189;;;12013:3;;;;12245:9;12238:5;12234:21;12229:2;1
;12831:6;12839;12847;12900:2;12888:9;12879:7;12875:23;12871:32;12868:52;;;12916:1;12913;12906:12;12868:52;12948:9;12942:16;12967:31;12992:5;12967:31;:::i;:::-
;13067:2;13052:18;;13046:25;13017:5;;-1:-1:-1;13080:33:28;13046:25;13080:33;:::i;:::-;13132:7;13122:17;;;13179:2;13168:9;13164:18;13158:25;13148:35;;12727:462;;;;;:::o;13194:127::-
;13255:10;13250:3;13246:20;13243:1;13236:31;13286:4;13283:1;13276:15;13310:4;13307:1;13300:15;13326:127;13387:10;13382:3;13378:20;13375:1;13368:31;13418:4;13415:1;13408:15;13442:4;13
;14068:9;;;14099;;14116:15;;;14110:22;;14096:37;14086:71;;14137:18;;::::i;14168:217::-
;14208:1;14234;14224:132;;14278:10;14273:3;14269:20;14266:1;14259:31;14313:4;14310:1;14303:15;14341:4;14338:1;14331:15;14224:132;-1:-1:-1;14370:9:28;;14168:217::o;14390:128::-
;14457:9;;;14478:11;;;14475:37;;;14492:18;;::::i;14523:188::-;14602:13;;-1:-1:-1;;;;;;14644:42:28;;14634:53;;14624:81;;14701:1;14698;14691:12;14624:81;14523:188;;;;:::o;14716:450::-
;14803:6;14811;14819;14872:2;14860:9;14851:7;14847:23;14843:32;14840:52;;;14888:1;14885;14878:12;14840:52;14911:40;14941:9;14911:40;:::i;:::-
;14901:50;;14970:49;15015:2;15004:9;15000:18;14970:49;:::i;:::-
;14960:59;;15062:2;15051:9;15047:18;15041:25;15106:10;15099:5;15095:22;15088:5;15085:33;15075:61;;15132:1;15129;15122:12;15075:61;15155:5;15145:15;;;14716:450;;;;;:::o;15171:251::-
;15241:6;15294:2;15282:9;15273:7;15269:23;15265:32;15262:52;;;15310:1;15307;15300:12;15262:52;15342:9;15336:16;15361:31;15386:5;15361:31;:::i", "linkReferences": {}, "immutableReferences":
{ "31558": [{ "start": 311, "length": 32 }, { "start": 913, "length": 32 }, { "start": 1111, "length": 32 }, { "start": 1752, "length": 32 }, { "start": 2005, "length": 32 }, { "start": 2161, "length": 32 }, { "start": 2900,
"length": 32 }, { "start": 3046, "length": 32 }, { "start": 3384, "length": 32 }, { "start": 4053, "length": 32 }] } }, "methodIdentifiers": { "WETH_ADDRESS()": "040141e5", "call(address,uint256,bytes)":
"6dbf2fa0", "executeArbitrage(address,address,uint256,address)": "00fb4304", "getAmountIn((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))": "eb6b2116",
"getDenominator((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))": "1fa7d39b", "getNumerator((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))": "00bb7d67",
"makeFlashLoan(address[],uint256[],bytes)": "c9a69562", "owner()": "8da5cb5b", "receiveFlashLoan(address[],uint256[],uint256[],bytes)": "f04f2707", "renounceOwnership()": "715018a6",
"transferOwnership(address)": "f2fde38b", "withdrawETHToOwner()": "77432b8c", "withdrawWETHToOwner()": "ea3e506c" }, "rawMetadata": "{\"compiler\":
{\"version\":\"0.8.19+commit.7dd6d404\"},\"language\":\"Solidity\",\"output\":{\"abi\":[{\"inputs\":
[{\"internalType\":\"address\",\"name\":\"_wethAddress\",\"type\":\"address\"}],\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"address\",\"name\":\"previousOwner\",\"type\":\"address\"},
{\"indexed\":true,\"internalType\":\"address\",\"name\":\"newOwner\",\"type\":\"address\"}],\"name\":\"OwnershipTransferred\",\"type\":\"event\"},{\"inputs\":[],\"name\":\"WETH_ADDRESS\",\"outputs\":
[{\"internalType\":\"address\",\"name\":\"\",\"type\":\"address\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"address payable\",\"name\":\"_to\",\"type\":\"address\"},
{\"internalType\":\"uint256\",\"name\":\"_value\",\"type\":\"uint256\"},{\"internalType\":\"bytes\",\"name\":\"_data\",\"type\":\"bytes\"}],\"name\":\"call\",\"outputs\":

```
[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"address\",\"name\":\"firstPairAddress\",\"type\":\"address\"},
{\"internalType\":\"address\",\"name\":\"secondPairAddress\",\"type\":\"address\"},{\"internalType\":\"uint256\",\"name\":\"percentageToPayToCoinbase\",\"type\":\"uint256\"},
{\"internalType\":\"address\",\"name\":\"recipient\",\"type\":\"address\"}],\"name\":\"executeArbitrage\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[{\"components\":
[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"firstPairData\",\"type\":\"tuple\"},{\"components\":[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},
{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"secondPairData\",\"type\":\"tuple\"}],\"name\":\"getAmountIn\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"components\":
[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"firstPairData\",\"type\":\"tuple\"},{\"components\":[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},
{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"secondPairData\",\"type\":\"tuple\"}],\"name\":\"getDenominator\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"components\":
[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"firstPairData\",\"type\":\"tuple\"},{\"components\":[{\"internalType\":\"uint256\",\"name\":\"reserve0\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"reserve1\",\"type\":\"uint256\"},{\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"},
{\"internalType\":\"bool\",\"name\":\"isWETHZero\",\"type\":\"bool\"}],\"internalType\":\"struct
IPairReserves.PairReserves\",\"name\":\"secondPairData\",\"type\":\"tuple\"}],\"name\":\"getNumerator\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"contract IERC20[]\",\"name\":\"tokens\",\"type\":\"address[]\"},
{\"internalType\":\"uint256[]\",\"name\":\"amounts\",\"type\":\"uint256[]\"},{\"internalType\":\"bytes\",\"name\":\"userData\",\"type\":\"bytes\"}],\"name\":\"makeFlashLoan\",\"outputs\":
[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[],\"name\":\"owner\",\"outputs\":
[{\"internalType\":\"address\",\"name\":\"\",\"type\":\"address\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"contract IERC20[]\",\"name\":\"tokens\",\"type\":\"address[]\"},
{\"internalType\":\"uint256[]\",\"name\":\"amounts\",\"type\":\"uint256[]\"},{\"internalType\":\"uint256[]\",\"name\":\"feeAmounts\",\"type\":\"uint256[]\"},
{\"internalType\":\"bytes\",\"name\":\"userData\",\"type\":\"bytes\"}],\"name\":\"receiveFlashLoan\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[],\"name\":\"renounceOwnership\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"address\",\"name\":\"newOwner\",\"type\":\"address\"}],\"name\":\"transferOwnership\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[],\"name\":\"withdrawETHToOwner\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[],\"name\":\"withdrawWETHToOwner\",\"outputs\":
[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"stateMutability\":\"payable\",\"type\":\"receive\"}],\"devdoc\":{\"kind\":\"dev\",\"methods\":{\"call(address,uint256,bytes)\":{\"details\":\"Only the
contract owner can call this function.Reverted calls will result in a revert.\",\"params\":{\"_data\":\"The calldata to send with the call.\",\"_to\":\"The address of the contract to call.\",\"_value\":\"The amount
of Ether to send with the call.\"}},\"executeArbitrage(address,address,uint256,address)\":{\"details\":\"Only the contract owner can call this function.\",\"params\":{\"firstPairAddress\":\"Address of the first
Uniswap V2 pair.\",\"secondPairAddress\":\"Address of the second Uniswap V2 pair.\"}},\"getAmountIn((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))\":{\"params\":
{\"firstPairData\":\"Struct containing data about the first Uniswap V2 pair.\",\"secondPairData\":\"Struct containing data about the second Uniswap V2 pair.\"},\"returns\":{\"_0\":\"amountIn, the optimal
amount to trade to arbitrage two v2 pairs.\"}},\"owner()\":{\"details\":\"Returns the address of the current owner.\"},\"receiveFlashLoan(address[],uint256[],uint256[],bytes)\":{\"details\":\"When flashLoan is
called on the Vault, it invokes the receiveFlashLoan hook on the recipient. At the time of the call, the Vault will have transferredamounts for tokens to the recipient. Before this call returns, the recipient must
have transferred amounts plus feeAmounts for each token back to the Vault, or else the entire flash loan will revertuserData is the same value passed in theIVault.flashLoan call.\"},\"renounceOwnership()\":
{\"details\":\"Leaves the contract without owner. It will not be possible to call onlyOwner functions. Can only be called by the current owner. NOTE: Renouncing ownership will leave the contract without an
owner, thereby disabling any functionality that is only available to the owner.\"},\"transferOwnership(address)\":{\"details\":\"Transfers ownership of the contract to a new account (newOwner). Can only be
called by the current owner.\"},\"withdrawETHToOwner()\":{\"details\":\"Only the contract owner can call this function.\"},\"withdrawWETHToOwner()\":{\"details\":\"Only the contract owner can call this
function.\"}},\"version\":1},\"userdoc\":{\"kind\":\"user\",\"methods\":{\"call(address,uint256,bytes)\":{\"notice\":\"Executes a call to another contract with the provided data and
value.\"},\"executeArbitrage(address,address,uint256,address)\":{\"notice\":\"Executes an arbitrage transaction between two Uniswap V2 pairs.Pair addresses need to be computed off-
chain.\"},\"getAmountIn((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))\":{\"notice\":\"Calculates the required input amount for the arbitrage transaction.\"},\"withdrawETHToOwner()\":
{\"notice\":\"Transfers all ETH held by the contract to the contract owner.\"},\"withdrawWETHToOwner()\":{\"notice\":\"Transfers all WETH held by the contract to the contract
owner.\"}},\"version\":1}},\"settings\":{\"compilationTarget\":{\"src/blindBackrunFL.sol\":\"BlindBackrunFL\"},\"evmVersion\":\"london\",\"libraries\":{},\"metadata\":{\"bytecodeHash\":\"ipfs\"},\"optimizer\":
{\"enabled\":true,\"runs\":200},\"remappings\":[\":ds-test/=lib/openzeppelin-contracts/lib/forge-std/lib/ds-test/src/\",\":erc4626-tests/=lib/openzeppelin-contracts/lib/erc4626-tests/\",\":forge-
std/=lib/openzeppelin-contracts/lib/forge-std/src/\",\":openzeppelin-contracts/=lib/openzeppelin-contracts/\",\":openzeppelin/=lib/openzeppelin-contracts/contracts/\"]},\"sources\":{\"lib/openzeppelin-
contracts/contracts/access/Ownable.sol\":{\"keccak256\":\"0x923b9774b81c1abfb992262ae7763b6e6de77b077a7180d53c6ebb7b1c8bd648\",\"license\":\"MIT\",\"urls\":[\"bzz-
raw://53445dc0431f9b45c06f567c6091da961d4087bec0010cca5bd62100fa624a38\",\"dweb:/ipfs/QmNvBYpBv183czrAqNXr76E8M3LF93ouAJFeAcHfb59Rcx\"]},\"lib/openzeppelin-
contracts/contracts/token/ERC20/IERC20.sol\":{\"keccak256\":\"0x00c839ff53d07d19db2e7cfa1e5133f9ee90a8d64b0e2e57f50446a2d1a3a0e0\",\"license\":\"MIT\",\"urls\":[\"bzz-
raw://3dac621d015a68a5251b1e5d41dda0faf252699bf6e8bcf46a958b29964d9dd1\",\"dweb:/ipfs/QmP9axjgZv4cezAhALoTemM62sdLtMDJ9MGTxECnNwHgSJ\"]},\"lib/openzeppelin-
contracts/contracts/utils/Context.sol\":{\"keccak256\":\"0xe2e337e6dde9ef6b680e07338c493ebea1b5fd09b43424112868e9cc1706bca7\",\"license\":\"MIT\",\"urls\":[\"bzz-
raw://6df0ddf21ce9f58271bdfaa85cde98b200ef242a05a3f85c2bc10a8294800a92\",\"dweb:/ipfs/QmRK2Y5Yc6BK7tGKkgsgn3aJEQGi5aakeSPZvS65PV8Xp3\"]},\"lib/openzeppelin-
contracts/contracts/utils/math/SafeMath.sol\":{\"keccak256\":\"0xa80cb30fccbba7c8ff72ac82cc898576a78d4c3031e85676840048f3fe195f5d\",\"license\":\"MIT\",\"urls\":[\"bzz-
raw://22fff2b2370631317e0e4bf36eabb136e96f1db5d5993c61e13b27d7615b504f\",\"dweb:/ipfs/QmfTW395az3LUhe3Gm5BUCwkaj9Sk1fURZKvHL2XzDHYQk\"]},\"src/BlindBackrun.sol\":
{\"keccak256\":\"0x94cb76a0a711d8725db99bf728521638548ae444077d4485e3a23fb87f4d87da\",\"urls\":[\"bzz-
raw://bbe803026bb99cb6f926f19a5d8fa50c8ea7cee30f49fd58df906c8cc91b7aa1\",\"dweb:/ipfs/QmSynZeD4hv25hH9e5ZgFzB78WbxYjpza3gRPrrPanenAo\"]},\"src/blindBackrunFL.sol\":
{\"keccak256\":\"0xd823a06388208db5cfdf08f3d6a87ca26876777317dd721037a845d4e25ffd4f\",\"urls\":[\"bzz-
raw://9e07a30427d99d1a3dae21d358f49fb1f1eee7bbf217854fa4dd741d595ecd7b\",\"dweb:/ipfs/Qmd2MVgu5qRd1Yuub2rUdQg1F61MKBX6ZHUCmaGbsDxL97\"]}},\"version\":1}", "metadata": {
"compiler": { "version": "0.8.19+commit.7dd6d404" }, "language": "Solidity", "output": { "abi": [ { "inputs": [ { "internalType": "address", "name": "_wethAddress", "type": "address" } ], "stateMutability":
"nonpayable", "type": "constructor" }, { "inputs": [ { "internalType": "address", "name": "previousOwner", "type": "address", "indexed": true }, { "internalType": "address", "name": "newOwner", "type":
"address", "indexed": true } ], "type": "event", "name": "OwnershipTransferred", "anonymous": false }, { "inputs": [], "stateMutability": "view", "type": "function", "name": "WETH_ADDRESS", "outputs": [ {
"internalType": "address", "name": "", "type": "address" } ] }, { "inputs": [ { "internalType": "address payable", "name": "_to", "type": "address" }, { "internalType": "uint256", "name": "_value", "type":
"uint256" }, { "internalType": "bytes", "name": "_data", "type": "bytes" } ], "stateMutability": "nonpayable", "type": "function", "name": "call" }, { "inputs": [ { "internalType": "address", "name":
"firstPairAddress", "type": "address" }, { "internalType": "address", "name": "secondPairAddress", "type": "address" }, { "internalType": "uint256", "name": "percentageToPayToCoinbase", "type":
"uint256" }, { "internalType": "address", "name": "recipient", "type": "address" } ], "stateMutability": "nonpayable", "type": "function", "name": "executeArbitrage" }, { "inputs": [ { "internalType": "struct
IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple", "components": [ { "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1",
"type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" } ] }, { "internalType": "struct IPairReserves.PairReserves",
"name": "secondPairData", "type": "tuple", "components": [ { "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, {
"internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" } ] } ], "stateMutability": "view", "type": "function", "name": "getAmountIn",
"outputs": [ { "internalType": "uint256", "name": "", "type": "uint256" } ] }, { "inputs": [ { "internalType": "struct IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple", "components": [ {
"internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, {
"internalType": "bool", "name": "isWETHZero", "type": "bool" } ] }, { "internalType": "struct IPairReserves.PairReserves", "name": "secondPairData", "type": "tuple", "components": [ { "internalType":
"uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool",
"name": "isWETHZero", "type": "bool" } ] } ], "stateMutability": "view", "type": "function", "name": "getDenominator", "outputs": [ { "internalType": "uint256", "name": "", "type": "uint256" } ] }, { "inputs": [ {
"internalType": "struct IPairReserves.PairReserves", "name": "firstPairData", "type": "tuple", "components": [ { "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256",
"name": "reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" } ] }, { "internalType": "struct
IPairReserves.PairReserves", "name": "secondPairData", "type": "tuple", "components": [ { "internalType": "uint256", "name": "reserve0", "type": "uint256" }, { "internalType": "uint256", "name":
"reserve1", "type": "uint256" }, { "internalType": "uint256", "name": "price", "type": "uint256" }, { "internalType": "bool", "name": "isWETHZero", "type": "bool" } ] } ], "stateMutability": "view", "type":
"function", "name": "getNumerator", "outputs": [ { "internalType": "uint256", "name": "", "type": "uint256" } ] }, { "inputs": [ { "internalType": "contract IERC20[]", "name": "tokens", "type": "address[]" }, {
"internalType": "uint256[]", "name": "amounts", "type": "uint256[]" }, { "internalType": "bytes", "name": "userData", "type": "bytes" } ], "stateMutability": "nonpayable", "type": "function", "name":
"makeFlashLoan" }, { "inputs": [], "stateMutability": "view", "type": "function", "name": "owner", "outputs": [ { "internalType": "address", "name": "", "type": "address" } ] }, { "inputs": [ { "internalType":
"contract IERC20[]", "name": "tokens", "type": "address[]" }, { "internalType": "uint256[]", "name": "amounts", "type": "uint256[]" }, { "internalType": "uint256[]", "name": "feeAmounts", "type": "uint256[]" }, {
"internalType": "bytes", "name": "userData", "type": "bytes" } ], "stateMutability": "nonpayable", "type": "function", "name": "receiveFlashLoan" }, { "inputs": [], "stateMutability": "nonpayable", "type":
"function", "name": "renounceOwnership" }, { "inputs": [ { "internalType": "address", "name": "newOwner", "type": "address" } ], "stateMutability": "nonpayable", "type": "function", "name":
"transferOwnership" }, { "inputs": [], "stateMutability": "nonpayable", "type": "function", "name": "withdrawETHToOwner" }, { "inputs": [], "stateMutability": "nonpayable", "type": "function", "name":
"withdrawWETHToOwner" }, { "inputs": [], "stateMutability": "payable", "type": "receive" } ], "devdoc": { "kind": "dev", "methods": { "call(address,uint256,bytes)": { "details": "Only the contract owner can
call this function.Reverted calls will result in a revert.", "params": { "_data": "The calldata to send with the call.", "_to": "The address of the contract to call.", "_value": "The amount of Ether to send with
the call." } }, "executeArbitrage(address,address,uint256,address)": { "details": "Only the contract owner can call this function.", "params": { "firstPairAddress": "Address of the first Uniswap V2 pair.",
"secondPairAddress": "Address of the second Uniswap V2 pair." } }, "getAmountIn((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))": { "params": { "firstPairData": "Struct containing data
about the first Uniswap V2 pair.", "secondPairData": "Struct containing data about the second Uniswap V2 pair." }, "returns": { "_0": "amountIn, the optimal amount to trade to arbitrage two v2 pairs." } },
"owner()": { "details": "Returns the address of the current owner." }, "receiveFlashLoan(address[],uint256[],uint256[],bytes)": { "details": "When flashLoan is called on the Vault, it invokes thereceiveFlashLoan
```

hook on the recipient. At the time of the call, the Vault will have transferred amounts for tokens to the recipient. Before this call returns, the recipient must have transferredamounts plus feeAmounts for each
token back to the Vault, or else the entire flash loan will revert. userData is the same value passed in theIVault.flashLoan call." }, "renounceOwnership()": { "details": "Leaves the contract without owner. It
will not be possible to call onlyOwner functions. Can only be called by the current owner. NOTE: Renouncing ownership will leave the contract without an owner, thereby disabling any functionality that is
only available to the owner." }, "transferOwnership(address)": { "details": "Transfers ownership of the contract to a new account (newOwner). Can only be called by the current owner." },
"withdrawETHToOwner()": { "details": "Only the contract owner can call this function." }, "withdrawWETHToOwner()": { "details": "Only the contract owner can call this function." } }, "version": 1 },
"userdoc": { "kind": "user", "methods": { "call(address,uint256,bytes)": { "notice": "Executes a call to another contract with the provided data and value." },
"executeArbitrage(address,address,uint256,address)": { "notice": "Executes an arbitrage transaction between two Uniswap V2 pairs.Pair addresses need to be computed off-chain." },
"getAmountIn((uint256,uint256,uint256,bool),(uint256,uint256,uint256,bool))": { "notice": "Calculates the required input amount for the arbitrage transaction." }, "withdrawETHToOwner()": { "notice":
"Transfers all ETH held by the contract to the contract owner." }, "withdrawWETHToOwner()": { "notice": "Transfers all WETH held by the contract to the contract owner." } }, "version": 1 } }, "settings": {
"remappings": [":ds-test/=lib/openzeppelin-contracts/lib/forge-std/lib/ds-test/src/", ":erc4626-tests/=lib/openzeppelin-contracts/lib/erc4626-tests/", ":forge-std/=lib/openzeppelin-contracts/lib/forge-
std/src/", ":openzeppelin-contracts/=lib/openzeppelin-contracts/", ":openzeppelin/=lib/openzeppelin-contracts/contracts/"], "optimizer": { "enabled": true, "runs": 200 }, "metadata": { "bytecodeHash":
"ipfs" }, "compilationTarget": { "src/blindBackrunFL.sol": "BlindBackrunFL" }, "libraries": {} }, "sources": { "lib/openzeppelin-contracts/contracts/access/Ownable.sol": { "keccak256":
"0x923b9774b81c1abfb992262ae7763b6e6de77b077a7180d53c6ebb7b1c8bd648", "urls": ["bzz-raw://53445dc0431f9b45c06f567c6091da961d4087bec0010cca5bd62100fa624a38",
"dweb:/ipfs/QmNvBYpBv183czrAqNXr76E8M3LF93ouAJFeAcHfb59Rcx"], "license": "MIT" }, "lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol": { "keccak256":
"0x00c839ff53d07d19db2e7cfa1e5133f9ee90a8d64b0e2e57f50446a2d1a3a0e0", "urls": ["bzz-raw://3dac621d015a68a5251b1e5d41dda0faf252699bf6e8bcf46a958b29964d9dd1",
"dweb:/ipfs/QmP9axjgZv4cezAhALoTemM62sdLtMDJ9MGTxECnNwHgSJ"], "license": "MIT" }, "lib/openzeppelin-contracts/contracts/utils/Context.sol": { "keccak256":
"0xe2e337e6dde9ef6b680e07338c493ebea1b5fd09b43424112868e9cc1706bca7", "urls": ["bzz-raw://6df0ddf21ce9f58271bdfaa85cde98b200ef242a05a3f85c2bc10a8294800a92",
"dweb:/ipfs/QmRK2Y5Yc6BK7tGKkgsgn3aJEQGi5aakeSPZvS65PV8Xp3"], "license": "MIT" }, "lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol": { "keccak256":
"0xa80cb30fccbba7c8ff72ac82cc898576a78d4c3031e85676840048f3fe195f5d", "urls": ["bzz-raw://22fff2b2370631317e0e4bf36eabb136e96f1db5d5993c61e13b27d7615b504f",
"dweb:/ipfs/QmfTW395az3LUhe3Gm5BUCwkaj9Sk1fURZKvHL2XzDHYQk"], "license": "MIT" }, "src/BlindBackrun.sol": { "keccak256":
"0x94cb76a0a711d8725db99bf728521638548ae444077d4485e3a23fb87f4d87da", "urls": ["bzz-raw://bbe803026bb99cb6f926f19a5d8fa50c8ea7cee30f49fd58df906c8cc91b7aa1",
"dweb:/ipfs/QmSynZeD4hv25hH9e5ZgFzB78WbxYjpza3gRPrrPanenAo"], "license": null }, "src/blindBackrunFL.sol": { "keccak256":
"0xd823a06388208db5cfdf08f3d6a87ca26876777317dd721037a845d4e25ffd4f", "urls": ["bzz-raw://9e07a30427d99d1a3dae21d358f49fb1f1eee7bbf217854fa4dd741d595ecd7b",
"dweb:/ipfs/Qmd2MVgu5qRd1Yuub2rUdQg1F61MKBX6ZHUCmaGbsDxL97"], "license": null } }, "version": 1 }, "ast": { "absolutePath": "src/blindBackrunFL.sol", "id": 30460, "exportedSymbols": {
"BlindBackrun": [32259], "BlindBackrunFL": [30459], "Context": [824], "IERC20": [777], "IFlashLoanRecipient": [30342], "IPairReserves": [31548], "IUniswapV2Pair": [31538], "IVault": [30325],
"IWETH": [31507], "Ownable": [112], "SafeMath": [1136] }, "nodeType": "SourceUnit", "src": "0:2345:26", "nodes": [{ "id": 30305, "nodeType": "PragmaDirective", "src": "0:23:26", "nodes": [],
"literals": ["solidity", "^", "0.8", ".0"] }, { "id": 30306, "nodeType": "ImportDirective", "src": "25:45:26", "nodes": [], "absolutePath": "lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol", "file":
"openzeppelin/token/ERC20/IERC20.sol", "nameLocation": "-1:-1:-1", "scope": 30460, "sourceUnit": 778, "symbolAliases": [], "unitAlias": "" }, { "id": 30307, "nodeType": "ImportDirective", "src":
"71:41:26", "nodes": [], "absolutePath": "lib/openzeppelin-contracts/contracts/access/Ownable.sol", "file": "openzeppelin/access/Ownable.sol", "nameLocation": "-1:-1:-1", "scope": 30460, "sourceUnit":
113, "symbolAliases": [], "unitAlias": "" }, { "id": 30308, "nodeType": "ImportDirective", "src": "113:46:26", "nodes": [], "absolutePath": "lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol", "file":
"openzeppelin/utils/math/SafeMath.sol", "nameLocation": "-1:-1:-1", "scope": 30460, "sourceUnit": 1137, "symbolAliases": [], "unitAlias": "" }, { "id": 30309, "nodeType": "ImportDirective", "src":
"160:28:26", "nodes": [], "absolutePath": "src/BlindBackrun.sol", "file": "./BlindBackrun.sol", "nameLocation": "-1:-1:-1", "scope": 30460, "sourceUnit": 32260, "symbolAliases": [], "unitAlias": "" }, { "id":
30325, "nodeType": "ContractDefinition", "src": "190:195:26", "nodes": [{ "id": 30324, "nodeType": "FunctionDefinition", "src": "213:170:26", "nodes": [], "functionSelector": "5c38449e", "implemented":
false, "kind": "function", "modifiers": [], "name": "flashLoan", "nameLocation": "222:9:26", "parameters": { "id": 30322, "nodeType": "ParameterList", "parameters": [{ "constant": false, "id": 30312,
"mutability": "mutable", "name": "recipient", "nameLocation": "261:9:26", "nodeType": "VariableDeclaration", "scope": 30324, "src": "241:29:26", "stateVariable": false, "storageLocation": "default",
"typeDescriptions": { "typeIdentifier": "t_contract_IFlashLoanRecipient_30342", "typeString": "contract IFlashLoanRecipient" }, "typeName": { "id": 30311, "nodeType": "UserDefinedTypeName",
"pathNode": { "id": 30310, "name": "IFlashLoanRecipient", "nameLocations": ["241:19:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 30342, "src": "241:19:26" }, "referencedDeclaration":
30342, "src": "241:19:26", "typeDescriptions": { "typeIdentifier": "t_contract_IFlashLoanRecipient_30342", "typeString": "contract IFlashLoanRecipient" } }, "visibility": "internal" }, { "constant": false, "id":
30316, "mutability": "mutable", "name": "tokens", "nameLocation": "296:6:26", "nodeType": "VariableDeclaration", "scope": 30324, "src": "280:22:26", "stateVariable": false, "storageLocation": "memory",
"typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[]" }, "typeName": { "baseType": { "id": 30314, "nodeType":
"UserDefinedTypeName", "pathNode": { "id": 30313, "name": "IERC20", "nameLocations": ["280:6:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 777, "src": "280:6:26" },
"referencedDeclaration": 777, "src": "280:6:26", "typeDescriptions": { "typeIdentifier": "t_contract_IERC20_777", "typeString": "contract IERC20" } }, "id": 30315, "nodeType": "ArrayTypeName", "src":
"280:8:26", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_storage_ptr", "typeString": "contract IERC20[]" } }, "visibility": "internal" }, { "constant": false, "id": 30319,
"mutability": "mutable", "name": "amounts", "nameLocation": "329:7:26", "nodeType": "VariableDeclaration", "scope": 30324, "src": "312:24:26", "stateVariable": false, "storageLocation": "memory",
"typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30317, "name": "uint256", "nodeType": "ElementaryTypeName",
"src": "312:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30318, "nodeType": "ArrayTypeName", "src": "312:9:26", "typeDescriptions": { "typeIdentifier":
"t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30321, "mutability": "mutable", "name": "userData", "nameLocation": "359:8:26",
"nodeType": "VariableDeclaration", "scope": 30324, "src": "346:21:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_bytes_memory_ptr", "typeString":
"bytes" }, "typeName": { "id": 30320, "name": "bytes", "nodeType": "ElementaryTypeName", "src": "346:5:26", "typeDescriptions": { "typeIdentifier": "t_bytes_storage_ptr", "typeString": "bytes" } },
"visibility": "internal" }], "src": "231:142:26" }, "returnParameters": { "id": 30323, "nodeType": "ParameterList", "parameters": [], "src": "382:0:26" }, "scope": 30325, "stateMutability": "nonpayable",
"virtual": false, "visibility": "external" }], "abstract": false, "baseContracts": [], "canonicalName": "IVault", "contractDependencies": [], "contractKind": "interface", "fullyImplemented": false,
"linearizedBaseContracts": [30325], "name": "IVault", "nameLocation": "200:6:26", "scope": 30460, "usedErrors": [] }, { "id": 30342, "nodeType": "ContractDefinition", "src": "387:714:26", "nodes": [{ "id":
30341, "nodeType": "FunctionDefinition", "src": "924:175:26", "nodes": [], "documentation": { "id": 30326, "nodeType": "StructuredDocumentation", "src": "423:496:26", "text": " @dev When flashLoan is
called on the Vault, it invokes the receiveFlashLoan hook on the recipient.\n At the time of the call, the Vault will have transferredamounts for tokens to the recipient. Before this\n call returns, the recipient
must have transferred amounts plus feeAmounts for each token back to the\n Vault, or else the entire flash loan will revert.\nuserData is the same value passed in theIVault.flashLoan call." },
"functionSelector": "f04f2707", "implemented": false, "kind": "function", "modifiers": [], "name": "receiveFlashLoan", "nameLocation": "933:16:26", "parameters": { "id": 30339, "nodeType":
"ParameterList", "parameters": [{ "constant": false, "id": 30330, "mutability": "mutable", "name": "tokens", "nameLocation": "975:6:26", "nodeType": "VariableDeclaration", "scope": 30341, "src":
"959:22:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[]" },
"typeName": { "baseType": { "id": 30328, "nodeType": "UserDefinedTypeName", "pathNode": { "id": 30327, "name": "IERC20", "nameLocations": ["959:6:26"], "nodeType": "IdentifierPath",
"referencedDeclaration": 777, "src": "959:6:26" }, "referencedDeclaration": 777, "src": "959:6:26", "typeDescriptions": { "typeIdentifier": "t_contract_IERC20_777", "typeString": "contract IERC20" } }, "id":
30329, "nodeType": "ArrayTypeName", "src": "959:8:26", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_storage_ptr", "typeString": "contract IERC20[]" } }, "visibility":
"internal" }, { "constant": false, "id": 30333, "mutability": "mutable", "name": "amounts", "nameLocation": "1008:7:26", "nodeType": "VariableDeclaration", "scope": 30341, "src": "991:24:26",
"stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30331,
"name": "uint256", "nodeType": "ElementaryTypeName", "src": "991:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30332, "nodeType": "ArrayTypeName", "src":
"991:9:26", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30336, "mutability": "mutable", "name":
"feeAmounts", "nameLocation": "1042:10:26", "nodeType": "VariableDeclaration", "scope": 30341, "src": "1025:27:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": {
"typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30334, "name": "uint256", "nodeType": "ElementaryTypeName", "src": "1025:7:26",
"typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30335, "nodeType": "ArrayTypeName", "src": "1025:9:26", "typeDescriptions": { "typeIdentifier":
"t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30338, "mutability": "mutable", "name": "userData", "nameLocation": "1075:8:26",
"nodeType": "VariableDeclaration", "scope": 30341, "src": "1062:21:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_bytes_memory_ptr", "typeString":
"bytes" }, "typeName": { "id": 30337, "name": "bytes", "nodeType": "ElementaryTypeName", "src": "1062:5:26", "typeDescriptions": { "typeIdentifier": "t_bytes_storage_ptr", "typeString": "bytes" } },
"visibility": "internal" }], "src": "949:140:26" }, "returnParameters": { "id": 30340, "nodeType": "ParameterList", "parameters": [], "src": "1098:0:26" }, "scope": 30342, "stateMutability": "nonpayable",
"virtual": false, "visibility": "external" }], "abstract": false, "baseContracts": [], "canonicalName": "IFlashLoanRecipient", "contractDependencies": [], "contractKind": "interface", "fullyImplemented": false,
"linearizedBaseContracts": [30342], "name": "IFlashLoanRecipient", "nameLocation": "397:19:26", "scope": 30460, "usedErrors": [] }, { "id": 30459, "nodeType": "ContractDefinition", "src":
"1103:1241:26", "nodes": [{ "id": 30352, "nodeType": "VariableDeclaration", "src": "1170:90:26", "nodes": [], "constant": true, "mutability": "constant", "name": "vault", "nameLocation": "1194:5:26",
"scope": 30459, "stateVariable": true, "storageLocation": "default", "typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" }, "typeName": { "id": 30348,
"nodeType": "UserDefinedTypeName", "pathNode": { "id": 30347, "name": "IVault", "nameLocations": ["1170:6:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 30325, "src": "1170:6:26" },
"referencedDeclaration": 30325, "src": "1170:6:26", "typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" } }, "value": { "arguments": [{ "hexValue":
"307842413132323232323232323232323238643842613434353935386137356130373034643536364246324338", "id": 30350, "isConstant": false, "isLValue": false, "isPure": true, "kind": "number",
"lValueRequested": false, "nodeType": "Literal", "src": "1217:42:26", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" }, "value":
"0xBA12222222228d8Ba445958a75a0704d566BF2C8" }], "expression": { "argumentTypes": [{ "typeIdentifier": "t_address", "typeString": "address" }], "id": 30349, "name": "IVault", "nodeType":
"Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30325, "src": "1210:6:26", "typeDescriptions": { "typeIdentifier": "t_type_t_contract_IVault_30325", "typeString": "type(contract IVault)" }
}, "id": 30351, "isConstant": false, "isLValue": false, "isPure": true, "kind": "typeConversion", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "1210:50:26",
"tryCall": false, "typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" } }, "visibility": "private" }, { "id": 30361, "nodeType": "FunctionDefinition", "src":
"1267:63:26", "nodes": [], "body": { "id": 30360, "nodeType": "Block", "src": "1328:2:26", "nodes": [], "statements": [] }, "implemented": true, "kind": "constructor", "modifiers": [{ "arguments": [{ "id":
30357, "name": "wethAddress", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30354, "src": "1314:12:26", "typeDescriptions": { "typeIdentifier": "t_address", "typeString":
"address" } }], "id": 30358, "kind": "baseConstructorSpecifier", "modifierName": { "id": 30356, "name": "BlindBackrun", "nameLocations": ["1301:12:26"], "nodeType": "IdentifierPath",
"referencedDeclaration": 32259, "src": "1301:12:26" }, "nodeType": "ModifierInvocation", "src": "1301:26:26" }], "name": "", "nameLocation": "-1:-1:-1", "parameters": { "id": 30355, "nodeType":
"ParameterList", "parameters": [{ "constant": false, "id": 30354, "mutability": "mutable", "name": "_wethAddress", "nameLocation": "1287:12:26", "nodeType": "VariableDeclaration", "scope": 30361,
"src": "1279:20:26", "stateVariable": false, "storageLocation": "default", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" }, "typeName": { "id": 30353, "name": "address",
"nodeType": "ElementaryTypeName", "src": "1279:7:26", "stateMutability": "nonpayable", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, "visibility": "internal" }], "src":
"1278:22:26" }, "returnParameters": { "id": 30359, "nodeType": "ParameterList", "parameters": [], "src": "1328:0:26" }, "scope": 30459, "stateMutability": "nonpayable", "virtual": false, "visibility": "public" },
{ "id": 30385, "nodeType": "FunctionDefinition", "src": "1336:210:26", "nodes": [], "body": { "id": 30384, "nodeType": "Block", "src": "1481:65:26", "nodes": [], "statements": [{ "expression": { "arguments": [
{ "id": 30378, "name": "this", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": -28, "src": "1507:4:26", "typeDescriptions": { "typeIdentifier":
"t_contract_BlindBackrunFL_30459", "typeString": "contract BlindBackrunFL" } }, { "id": 30379, "name": "tokens", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30365,

```
"src": "1513:6:26", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[] memory" } }, { "id": 30380, "name": "amounts", "nodeType":
"Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30368, "src": "1521:7:26", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[] memory"
} }, { "id": 30381, "name": "userData", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30370, "src": "1530:8:26", "typeDescriptions": { "typeIdentifier":
"t_bytes_memory_ptr", "typeString": "bytes memory" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_contract_BlindBackrunFL_30459", "typeString": "contract BlindBackrunFL" }, {
"typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[] memory" }, { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]
memory" }, { "typeIdentifier": "t_bytes_memory_ptr", "typeString": "bytes memory" } ], "expression": { "id": 30375, "name": "vault", "nodeType": "Identifier", "overloadedDeclarations": [],
"referencedDeclaration": 30352, "src": "1491:5:26", "typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" } }, "id": 30377, "isConstant": false, "isLValue": false,
"isPure": false, "lValueRequested": false, "memberLocation": "1497:9:26", "memberName": "flashLoan", "nodeType": "MemberAccess", "referencedDeclaration": 30324, "src": "1491:15:26",
"typeDescriptions": { "typeIdentifier":
"t_function_external_nonpayable_t_contract_IFlashLoanRecipient_30342__t_array_t_contract_IERC20_777_dyn_memory_ptr__t_array_t_uint256_dyn_memory_ptr__t_bytes_memory_ptr_returns__",
"typeString": "function (contract IFlashLoanRecipient,contract IERC20[] memory,uint256[] memory,bytes memory) external" } }, "id": 30382, "isConstant": false, "isLValue": false, "isPure": false, "kind":
"functionCall", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "1491:48:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_tuple__", "typeString":
"tuple()" } }, "id": 30383, "nodeType": "ExpressionStatement", "src": "1491:48:26" } ] }, "functionSelector": "c9a69562", "implemented": true, "kind": "function", "modifiers": [ { "id": 30373, "kind":
"modifierInvocation", "modifierName": { "id": 30372, "name": "onlyOwner", "nameLocations": [ "1471:9:26" ], "nodeType": "IdentifierPath", "referencedDeclaration": 31, "src": "1471:9:26" }, "nodeType":
"ModifierInvocation", "src": "1471:9:26" } ], "name": "makeFlashLoan", "nameLocation": "1345:13:26", "parameters": { "id": 30371, "nodeType": "ParameterList", "parameters": [ { "constant": false, "id":
30365, "mutability": "mutable", "name": "tokens", "nameLocation": "1384:6:26", "nodeType": "VariableDeclaration", "scope": 30385, "src": "1368:22:26", "stateVariable": false, "storageLocation":
"memory", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[]" }, "typeName": { "baseType": { "id": 30363, "nodeType":
"UserDefinedTypeName", "pathNode": { "id": 30362, "name": "IERC20", "nameLocations": [ "1368:6:26" ], "nodeType": "IdentifierPath", "referencedDeclaration": 777, "src": "1368:6:26" },
"referencedDeclaration": 777, "src": "1368:6:26", "typeDescriptions": { "typeIdentifier": "t_contract_IERC20_777", "typeString": "contract IERC20" } }, "id": 30364, "nodeType": "ArrayTypeName", "src":
"1368:8:26", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_storage_ptr", "typeString": "contract IERC20[]" } }, "visibility": "internal" }, { "constant": false, "id": 30368,
"mutability": "mutable", "name": "amounts", "nameLocation": "1417:7:26", "nodeType": "VariableDeclaration", "scope": 30385, "src": "1400:24:26", "stateVariable": false, "storageLocation": "memory",
"typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30366, "name": "uint256", "nodeType": "ElementaryTypeName",
"src": "1400:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30367, "nodeType": "ArrayTypeName", "src": "1400:9:26", "typeDescriptions": { "typeIdentifier":
"t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30370, "mutability": "mutable", "name": "userData", "nameLocation": "1447:8:26",
"nodeType": "VariableDeclaration", "scope": 30385, "src": "1434:21:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_bytes_memory_ptr", "typeString":
"bytes" }, "typeName": { "id": 30369, "name": "bytes", "nodeType": "ElementaryTypeName", "src": "1434:5:26", "typeDescriptions": { "typeIdentifier": "t_bytes_storage_ptr", "typeString": "bytes" } },
"visibility": "internal" } ], "src": "1358:103:26" }, "returnParameters": { "id": 30374, "nodeType": "ParameterList", "parameters": [], "src": "1481:0:26" }, "scope": 30459, "stateMutability": "nonpayable",
"virtual": false, "visibility": "external" }, { "id": 30458, "nodeType": "FunctionDefinition", "src": "1552:790:26", "nodes": [], "body": { "id": 30457, "nodeType": "Block", "src": "1727:615:26", "nodes": [],
"statements": [ { "expression": { "arguments": [ { "commonType": { "typeIdentifier": "t_address", "typeString": "address" }, "id": 30407, "isConstant": false, "isLValue": false, "isPure": false,
"lValueRequested": false, "leftExpression": { "expression": { "id": 30401, "name": "msg", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": -15, "src": "1758:3:26",
"typeDescriptions": { "typeIdentifier": "t_magic_message", "typeString": "msg" } }, "id": 30402, "isConstant": false, "isLValue": false, "isPure": false, "lValueRequested": false, "memberLocation":
"1762:6:26", "memberName": "sender", "nodeType": "MemberAccess", "src": "1758:10:26", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, "nodeType": "BinaryOperation",
"operator": "==", "rightExpression": { "arguments": [ { "id": 30405, "name": "vault", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30352, "src": "1780:5:26",
"typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract
IVault" } ], "id": 30404, "isConstant": false, "isLValue": false, "isPure": true, "lValueRequested": false, "nodeType": "ElementaryTypeNameExpression", "src": "1772:7:26", "typeDescriptions": {
"typeIdentifier": "t_type_t_address", "typeString": "type(address)" }, "typeName": { "id": 30403, "name": "address", "nodeType": "ElementaryTypeName", "src": "1772:7:26", "typeDescriptions": {} } }, "id":
30406, "isConstant": false, "isLValue": false, "isPure": true, "kind": "typeConversion", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "1772:14:26",
"tryCall": false, "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, "src": "1758:28:26", "typeDescriptions": { "typeIdentifier": "t_bool", "typeString": "bool" } }, { "hexValue":
"466c6173684c6f616e526563697069656e743a2063616c6c6572206973206e6f7420746865207661756c74", "id": 30408, "isConstant": false, "isLValue": false, "isPure": true, "kind": "string",
"lValueRequested": false, "nodeType": "Literal", "src": "1800:45:26", "typeDescriptions": { "typeIdentifier": "t_stringliteral_0f31b4f6d08b07b44b6c6f09cb866661e71ab79b49d3cbdc3ea1d936d7a195c5",
"typeString": "literal_string \"FlashLoanRecipient: caller is not the vault\"" }, "value": "FlashLoanRecipient: caller is not the vault" } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_bool",
"typeString": "bool" }, { "typeIdentifier": "t_stringliteral_0f31b4f6d08b07b44b6c6f09cb866661e71ab79b49d3cbdc3ea1d936d7a195c5", "typeString": "literal_string \"FlashLoanRecipient: caller is not the
vault\"" } ], "id": 30400, "name": "require", "nodeType": "Identifier", "overloadedDeclarations": [ -18, -18 ], "referencedDeclaration": -18, "src": "1737:7:26", "typeDescriptions": { "typeIdentifier":
"t_function_require_pure_t_bool__t_string_memory_ptr_returns__", "typeString": "function (bool,string memory) pure" } }, "id": 30409, "isConstant": false, "isLValue": false, "isPure": false, "kind":
"functionCall", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "1737:118:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_tuple__", "typeString":
"tuple()" } }, "id": 30410, "nodeType": "ExpressionStatement", "src": "1737:118:26" }, { "assignments": [ 30412, 30414, 30416 ], "declarations": [ { "constant": false, "id": 30412, "mutability": "mutable",
"name": "firstPairAddress", "nameLocation": "1888:16:26", "nodeType": "VariableDeclaration", "scope": 30457, "src": "1880:24:26", "stateVariable": false, "storageLocation": "default", "typeDescriptions":
{ "typeIdentifier": "t_address", "typeString": "address" }, "typeName": { "id": 30411, "name": "address", "nodeType": "ElementaryTypeName", "src": "1880:7:26", "stateMutability": "nonpayable",
"typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, "visibility": "internal" }, { "constant": false, "id": 30414, "mutability": "mutable", "name": "secondPairAddress", "nameLocation":
"1926:17:26", "nodeType": "VariableDeclaration", "scope": 30457, "src": "1918:25:26", "stateVariable": false, "storageLocation": "default", "typeDescriptions": { "typeIdentifier": "t_address", "typeString":
"address" }, "typeName": { "id": 30413, "name": "address", "nodeType": "ElementaryTypeName", "src": "1918:7:26", "stateMutability": "nonpayable", "typeDescriptions": { "typeIdentifier": "t_address",
"typeString": "address" } }, "visibility": "internal" }, { "constant": false, "id": 30416, "mutability": "mutable", "name": "percentageToPayToCoinbase", "nameLocation": "1965:25:26", "nodeType":
"VariableDeclaration", "scope": 30457, "src": "1957:33:26", "stateVariable": false, "storageLocation": "default", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" }, "typeName": {
"id": 30415, "name": "uint256", "nodeType": "ElementaryTypeName", "src": "1957:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "visibility": "internal" } ], "id": 30428,
"initialValue": { "arguments": [ { "id": 30419, "name": "userData", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30397, "src": "2014:8:26", "typeDescriptions": {
"typeIdentifier": "t_bytes_memory_ptr", "typeString": "bytes memory" } }, { "components": [ { "id": 30421, "isConstant": false, "isLValue": false, "isPure": true, "lValueRequested": false, "nodeType":
"ElementaryTypeNameExpression", "src": "2025:7:26", "typeDescriptions": { "typeIdentifier": "t_type_t_address_", "typeString": "type(address)" }, "typeName": { "id": 30420, "name": "address",
"nodeType": "ElementaryTypeName", "src": "2025:7:26", "typeDescriptions": {} } }, { "id": 30423, "isConstant": false, "isLValue": false, "isPure": true, "lValueRequested": false, "nodeType":
"ElementaryTypeNameExpression", "src": "2034:7:26", "typeDescriptions": { "typeIdentifier": "t_type_t_address_", "typeString": "type(address)" }, "typeName": { "id": 30422, "name": "address",
"nodeType": "ElementaryTypeName", "src": "2034:7:26", "typeDescriptions": {} } }, { "id": 30425, "isConstant": false, "isLValue": false, "isPure": true, "lValueRequested": false, "nodeType":
"ElementaryTypeNameExpression", "src": "2043:7:26", "typeDescriptions": { "typeIdentifier": "t_type_t_uint256_", "typeString": "type(uint256)" }, "typeName": { "id": 30424, "name": "uint256",
"nodeType": "ElementaryTypeName", "src": "2043:7:26", "typeDescriptions": {} } } ], "id": 30426, "isConstant": false, "isInlineArray": false, "isLValue": false, "isPure": true, "lValueRequested": false,
"nodeType": "TupleExpression", "src": "2024:27:26", "typeDescriptions": { "typeIdentifier": "t_tuple_t_type_t_addresstype_t_addresst_type_t_uint256__", "typeString":
"tuple(type(address),type(address),type(uint256))" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_bytes_memory_ptr", "typeString": "bytes memory" }, { "typeIdentifier":
"t_tuple_t_type_t_address___t_type_t_address___t_type_t_uint256__", "typeString": "tuple(type(address),type(address),type(uint256))" } ], "expression": { "id": 30417, "name": "abi", "nodeType":
"Identifier", "overloadedDeclarations": [], "referencedDeclaration": -1, "src": "2003:3:26", "typeDescriptions": { "typeIdentifier": "t_magic_abi", "typeString": "abi" } }, "id": 30418, "isConstant": false,
"isLValue": false, "isPure": true, "lValueRequested": false, "memberLocation": "2007:6:26", "memberName": "decode", "nodeType": "MemberAccess", "src": "2003:10:26", "typeDescriptions": {
"typeIdentifier": "t_function_abidecode_pure__returns__", "typeString": "function () pure" } }, "id": 30427, "isConstant": false, "isLValue": false, "isPure": false, "kind": "functionCall", "lValueRequested":
false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "2003:49:26", "tryCall": false, "typeDescriptions": { "typeIdentifier":
"t_tuple_t_address_payable__t_address_payable__t_uint256", "typeString": "tuple(address payable,address payable,uint256)" } }, "nodeType": "VariableDeclarationStatement", "src": "1866:186:26" }, {
"expression": { "arguments": [ { "id": 30430, "name": "secondPairAddress", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30414, "src": "2093:17:26", "typeDescriptions":
{ "typeIdentifier": "t_address", "typeString": "address" } }, { "id": 30431, "name": "firstPairAddress", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30412, "src":
"2124:16:26", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, { "id": 30432, "name": "percentageToPayToCoinbase", "nodeType": "Identifier", "overloadedDeclarations": [],
"referencedDeclaration": 30416, "src": "2154:25:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, { "arguments": [ { "id": 30435, "name": "this", "nodeType": "Identifier",
"overloadedDeclarations": [], "referencedDeclaration": -28, "src": "2201:4:26", "typeDescriptions": { "typeIdentifier": "t_contract_BlindBackrunFL_30459", "typeString": "contract BlindBackrunFL" } } ],
"expression": { "argumentTypes": [ { "typeIdentifier": "t_contract_BlindBackrunFL_30459", "typeString": "contract BlindBackrunFL" } ], "id": 30434, "isConstant": false, "isLValue": false, "isPure": true,
"lValueRequested": false, "nodeType": "ElementaryTypeNameExpression", "src": "2193:7:26", "typeDescriptions": { "typeIdentifier": "t_type_t_address_", "typeString": "type(address)" }, "typeName": {
"id": 30433, "name": "address", "nodeType": "ElementaryTypeName", "src": "2193:7:26", "typeDescriptions": {} } }, "id": 30436, "isConstant": false, "isLValue": false, "isPure": false, "kind":
"typeConversion", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "2193:13:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_address",
"typeString": "address" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_address", "typeString": "address" }, { "typeIdentifier": "t_address", "typeString": "address" }, { "typeIdentifier":
"t_uint256", "typeString": "uint256" }, { "typeIdentifier": "t_address", "typeString": "address" } ], "id": 30429, "name": "executeArbitrage", "nodeType": "Identifier", "overloadedDeclarations": [],
"referencedDeclaration": 31797, "src": "2063:16:26", "typeDescriptions": { "typeIdentifier": "t_function_internal_nonpayable_t_address__t_address__t_uint256__t_address_returns__", "typeString":
"function (address,address,uint256,address)" } }, "id": 30437, "isConstant": false, "isLValue": false, "isPure": false, "kind": "functionCall", "lValueRequested": false, "nameLocations": [], "names": [],
"nodeType": "FunctionCall", "src": "2063:153:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_tuple__", "typeString": "tuple()" } }, "id": 30438, "nodeType": "ExpressionStatement", "src":
"2063:153:26" }, { "expression": { "arguments": [ { "arguments": [ { "id": 30445, "name": "vault", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30352, "src": "2277:5:26",
"typeDescriptions": { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract IVault" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_contract_IVault_30325", "typeString": "contract
IVault" } ], "id": 30444, "isConstant": false, "isLValue": false, "isPure": true, "lValueRequested": false, "nodeType": "ElementaryTypeNameExpression", "src": "2269:7:26", "typeDescriptions": {
"typeIdentifier": "t_type_t_address_", "typeString": "type(address)" }, "typeName": { "id": 30443, "name": "address", "nodeType": "ElementaryTypeName", "src": "2269:7:26", "typeDescriptions": {} } }, "id":
30446, "isConstant": false, "isLValue": false, "isPure": true, "kind": "typeConversion", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "2269:14:26",
"tryCall": false, "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } }, { "commonType": { "typeIdentifier": "t_uint256", "typeString": "uint256" }, "id": 30454, "isConstant": false,
"isLValue": false, "isPure": false, "lValueRequested": false, "leftExpression": { "baseExpression": { "id": 30447, "name": "amounts", "nodeType": "Identifier", "overloadedDeclarations": [],
"referencedDeclaration": 30392, "src": "2297:7:26", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[] memory" } }, "id": 30449, "indexExpression": {
"hexValue": "30", "id": 30448, "isConstant": false, "isLValue": false, "isPure": true, "kind": "number", "lValueRequested": false, "nodeType": "Literal", "src": "2305:1:26", "typeDescriptions": {
"typeIdentifier": "t_rational_0_by_1", "typeString": "int_const 0" }, "value": "0" }, "isConstant": false, "isLValue": true, "isPure": false, "lValueRequested": false, "nodeType": "IndexAccess", "src":
"2297:10:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "nodeType": "BinaryOperation", "operator": "+", "rightExpression": { "components": [ { "baseExpression": { "id":
30450, "name": "feeAmounts", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 30395, "src": "2311:10:26", "typeDescriptions": { "typeIdentifier":
"t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[] memory" } }, "id": 30452, "indexExpression": { "hexValue": "30", "id": 30451, "isConstant": false, "isLValue": false, "isPure": true, "kind":
"number", "lValueRequested": false, "nodeType": "Literal", "src": "2322:1:26", "typeDescriptions": { "typeIdentifier": "t_rational_0_by_1", "typeString": "int_const 0" }, "value": "0" }, "isConstant": false,
"isLValue": true, "isPure": false, "lValueRequested": false, "nodeType": "IndexAccess", "src": "2311:13:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } } ], "id": 30453,
"isConstant": false, "isInlineArray": false, "isLValue": false, "isPure": false, "lValueRequested": false, "nodeType": "TupleExpression", "src": "2310:15:26", "typeDescriptions": { "typeIdentifier":
"t_uint256", "typeString": "uint256" } }, "src": "2297:28:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } } ], "expression": { "argumentTypes": [ { "typeIdentifier": "t_address",
"typeString": "address" }, { "typeIdentifier": "t_uint256", "typeString": "uint256" } ], "expression": { "arguments": [ { "id": 30440, "name": "WETH_ADDRESS", "nodeType": "Identifier",
"overloadedDeclarations": [], "referencedDeclaration": 31558, "src": "2233:12:26", "typeDescriptions": { "typeIdentifier": "t_address", "typeString": "address" } } ], "expression": { "argumentTypes": [ {
```

"typeIdentifier": "t_address", "typeString": "address" }], "id": 30439, "name": "IWETH", "nodeType": "Identifier", "overloadedDeclarations": [], "referencedDeclaration": 31507, "src": "2227:5:26", "typeDescriptions": { "typeIdentifier": "t_type_t_contract_IWETH_31507_", "typeString": "type(contract IWETH)" } }, "id": 30441, "isConstant": false, "isLValue": false, "isPure": false, "kind": "typeConversion", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "2227:19:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_contract_IWETH_31507", "typeString": "contract IWETH" } }, "id": 30442, "isConstant": false, "isLValue": false, "isPure": false, "lValueRequested": false, "memberLocation": "2247:8:26", "memberName": "transfer", "nodeType": "MemberAccess", "referencedDeclaration": 744, "src": "2227:28:26", "typeDescriptions": { "typeIdentifier": "t_function_external_nonpayable_t_address__t_uint256_returns_t_bool_", "typeString": "function (address,uint256) external returns (bool)" } }, "id": 30455, "isConstant": false, "isLValue": false, "isPure": false, "kind": "functionCall", "lValueRequested": false, "nameLocations": [], "names": [], "nodeType": "FunctionCall", "src": "2227:108:26", "tryCall": false, "typeDescriptions": { "typeIdentifier": "t_bool", "typeString": "bool" } }, "id": 30456, "nodeType": "ExpressionStatement", "src": "2227:108:26" }] }, "baseFunctions": [30341], "functionSelector": "f04f2707", "implemented": true, "kind": "function", "modifiers": [], "name": "receiveFlashLoan", "nameLocation": "1561:16:26", "parameters": { "id": 30398, "nodeType": "ParameterList", "parameters": [{ "constant": false, "id": 30389, "mutability": "mutable", "name": "tokens", "nameLocation": "1603:6:26", "nodeType": "VariableDeclaration", "scope": 30458, "src": "1587:22:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_memory_ptr", "typeString": "contract IERC20[]" }, "typeName": { "baseType": { "id": 30387, "nodeType": "UserDefinedTypeName", "pathNode": { "id": 30386, "name": "IERC20", "nameLocations": ["1587:6:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 777, "src": "1587:6:26" }, "referencedDeclaration": 777, "src": "1587:6:26", "typeDescriptions": { "typeIdentifier": "t_contract_IERC20_777", "typeString": "contract IERC20" } }, "id": 30388, "nodeType": "ArrayTypeName", "src": "1587:8:26", "typeDescriptions": { "typeIdentifier": "t_array_t_contract_IERC20_777_dyn_storage_ptr", "typeString": "contract IERC20[]" } }, "visibility": "internal" }, { "constant": false, "id": 30392, "mutability": "mutable", "name": "amounts", "nameLocation": "1636:7:26", "nodeType": "VariableDeclaration", "scope": 30458, "src": "1619:24:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30390, "name": "uint256", "nodeType": "ElementaryTypeName", "src": "1619:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30391, "nodeType": "ArrayTypeName", "src": "1619:9:26", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30395, "mutability": "mutable", "name": "feeAmounts", "nameLocation": "1670:10:26", "nodeType": "VariableDeclaration", "scope": 30458, "src": "1653:27:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_memory_ptr", "typeString": "uint256[]" }, "typeName": { "baseType": { "id": 30393, "name": "uint256", "nodeType": "ElementaryTypeName", "src": "1653:7:26", "typeDescriptions": { "typeIdentifier": "t_uint256", "typeString": "uint256" } }, "id": 30394, "nodeType": "ArrayTypeName", "src": "1653:9:26", "typeDescriptions": { "typeIdentifier": "t_array_t_uint256_dyn_storage_ptr", "typeString": "uint256[]" } }, "visibility": "internal" }, { "constant": false, "id": 30397, "mutability": "mutable", "name": "userData", "nameLocation": "1703:8:26", "nodeType": "VariableDeclaration", "scope": 30458, "src": "1690:21:26", "stateVariable": false, "storageLocation": "memory", "typeDescriptions": { "typeIdentifier": "t_bytes_memory_ptr", "typeString": "bytes" }, "typeName": { "id": 30396, "name": "bytes", "nodeType": "ElementaryTypeName", "src": "1690:5:26", "typeDescriptions": { "typeIdentifier": "t_bytes_storage_ptr", "typeString": "bytes" } }, "visibility": "internal" }], "src": "1577:140:26" }, "returnParameters": { "id": 30399, "nodeType": "ParameterList", "parameters": [], "src": "1727:0:26" }, "scope": 30459, "stateMutability": "nonpayable", "virtual": false, "visibility": "external" }], "abstract": false, "baseContracts": [{ "baseName": { "id": 30343, "name": "BlindBackrun", "nameLocations": ["1130:12:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 32259, "src": "1130:12:26" }, "id": 30344, "nodeType": "InheritanceSpecifier", "src": "1130:12:26" }, { "baseName": { "id": 30345, "name": "IFlashLoanRecipient", "nameLocations": ["1144:19:26"], "nodeType": "IdentifierPath", "referencedDeclaration": 30342, "src": "1144:19:26" }, "id": 30346, "nodeType": "InheritanceSpecifier", "src": "1144:19:26" }], "canonicalName": "BlindBackrunFL", "contractDependencies": [], "contractKind": "contract", "fullyImplemented": true, "linearizedBaseContracts": [30459, 30342, 32259, 112, 824], "name": "BlindBackrunFL", "nameLocation": "1112:14:26", "scope": 30460, "usedErrors": [] }] }, "id": 26} This function sends a backrun given two pair addresses and a pending tx hash from the Event Stream:

```
import ArbContractAbi from

".abi/blindBackrunFlashLoan.json"

const arbContractAddress =

"0xTODO" const arbContract =

new

Contract ( arbContractAddress , ArbContractAbi , provider )

async

function

tryBackrun ( startPair , endPair , txHash )

{ let blockNumber =

Number ( await

this . signer . provider . getBlockNumber ( ) ) console . log ( "Current block number:" , blockNumber ) console . log ( "Building bundles" )

let bundleTransactionOptions =

{ gasPrice :

( await

this . signer . provider . getGasPrice ( ) ) ,

// This is extremely naive. gasLimit : ethers . BigNumber . from ( 400000 ) , nonce :

await

this . signer . getTransactionCount ( ) , } const types =

[ 'address' , 'address' , 'uint256' ] const values =

[ startPair , endPair , this . percentageToKeep ] let params = Web3EthAbi . encodeParameters ( types , values )

let bundleTransaction =

await arbContract . populateTransaction . makeFlashLoan ( config . mainnetWETHAddress , ethers . BigNumber . from ( 10 ** 21 ) . toString ( ) , params , bundleTransactionOptions ) let bundle =

[ { hash : txHash } , { tx :

await

this . signer . signTransaction ( bundleTransaction ) , canRevert :

false } , ] let fullBundle =

{ inclusion :

{ block : blockNumber +

1 ,

// try to land in next block maxBlock : ethers . utils . hexValue ( blockNumber +

24 )

// Protect txs expire after 25 blocks } , body : bundle , }

return

await mevshare . sendBundle ( fullBundle )

// for the reader: also send another bundle that calls makeFlashLoan // with startPair and endPair in switched in values. }
```

We getblockNumber at the start to set the target block in our bundle params.bundleTransactionOptions is pretty self-explanatory -- we use this because we're signing our backrun transaction manually, so it needs these params to be set manually.

types andvalues define the function params that are passed viauserData (as the variableparams) inmakeFlashLoan , and then decoded inreceiveFlashLoan and passed to_executeArbitrage .

We usearbContract.populateTransaction.makeFlashLoan to construct the backrun transaction, and then build a bundle with it and the user's pending tx hash. Lastly, we add the full bundle params for the MEV-SharesendBundle call, and then call it and return its result. mev-share-rs

// rs // TODO: Explain specific rust code here. Put these examples together and you should have a working flash-loan arbitrage bot. If you get stuck, checkout simple-blind-arbitrage , written in javascript. Edit this page Last updatedonFeb 13, 2024 Previous Flash Loan Basics (Balancer) Next April 2023 (beta launch)