But first, some Sushi

Part 2 for our Miniseries of articles leading up to D-Day

SushiGuard OpenMEV Router V03

This documents the finalized featureset for the router contract. Auditing and a bug bounty are to start tomorrow and be announced later this week (respectively).

New features

- Sushi Guard now sources flashloan liquidity from Bentobox's. This means more trades can be flashloaned, more of the flashloan fee stays within Sushiswap, and we save gas.
- No worse execution: Trades will be cheaper than existing trades. Arbitraged trade's are cheaper than existing trades with rebates from flashloaned liquidity.
- Subgraph now supports user tracking of earned rebates from profits. This is useful for tracking your rebated trades over time.

Walkthrough

Process is as follows. (using example 40 ETH → USDT)

If router has enough `balance` for optimal arb, use `balance`. (gas ~ 180,000, fee = 0.00%) If router does not have enough balance for optimal arb, use `bento` *if it* has sufficient balance. (gas ~ 230,000, fee = 0.05%) If router and`bento` do not have enough balance for optimal arb, use `aave`. (gas ~ 380,000, fee = 0.09%)

[

1176×732 15 KB

](https://global.discourse-cdn.com/standard10/uploads/manifold/original/1X/6d962b7d999782af21de83bf8a665567a78d88a7.png)

That's an example for a user swap from ETH → USDT in current market conditions

Graph shows MEV

extracted, without fees

(Flashloan fees can take away from the MEV profit)

User gas fees would be fixed at:

~90,000 for swap less than 0.1% reserve ~180,000 for swap where router has sufficient USDT to arb ~230,000 for swap where router does not have sufficient USDT to arb but `BentoBox` does ~390,000 for swap where router does not have sufficient USDT to arb nor `BentoBox` but `Aave` does

Subgraph and User data

MEV event

to store the protocol profit, token and user address from the trade

You should be able to verify and calculate user rebates (without fees) from the MEV profit.

emit MEV(user: 0xb4c79dab8f259c7aee6e5b2aa729821864227e84, token: 0x6b175474e89094c44da98b954eedeac495271d0f, value: 744148988356071006123)

We should be able to pick up and aggregate this information easily from the (Sushi Guard) subgraph

References

Graph Table

amountIn (eth)

amountOut (usdt)

Ca

Cb

Cf

Cg

arb amount (usdt)

mev (usdt)

0.1

243.2540764

7.29427E+17

7.20543E+17

8.88404E+15

37247288290

118892.23

0.00

1

2432.348157

8.67381E+17

7.20543E+17

1.46838E+17

37248182898

1879747.46

0.20

10

24304.23764

2.24572E+18

7.20543E+17

1.52518E+18

37257128979

14803014.79

21.17

20

48565.78235

3.77465E+18

7.20543E+17

3.05411E+18

37267069069

24918483.10

84.77

30

72784.74653

5.3009E+18

7.20543E+17

4.58036E+18

37277009159

33098644.07

190.70

40

96961.24217

6.82448E+18

7.20542E+17

6.10394E+18

37286949249

40147072.44

338.85

50

121095.3809

8.34538E+18

7.20542E+17

7.62484E+18

37296889339

46428544.82

529.10

60

145187.2739

9.86362E+18

7.20542E+17

9.14308E+18

37306829429

52145481.44

761.35

70

169237.032

1.13792E+19

7.20542E+17

1.06587E+19

37316769519

57424097.36

1035.48

80

193244.7657

1.28921E+19

7.20541E+17

1.21716E+19

37326709609

62349403.21

1351.38

90

217210.5849

1.44024E+19

7.20541E+17

1.36819E+19

37336649699

66981901.70

1708.95

100

241134.5995

1.59101E+19

7.20541E+17

1.51896E+19

37346589789

71366476.13

2108.07

BentoBox Interface

NOTE. it’s FlashLoan

not Flasloan

function testFlash(address token, uint256 amount) external { bentoBox.flashLoan(IFlashBorrower(address(this)), address(this), IERC20(token), amount, bytes("0")); }

```
function onFlashLoan(
    address sender,
    IERC20 token,
    uint256 amount,
    uint256 fee,
    bytes calldata data
) external {
    uint256 amountToReturn = amount + fee;
    token.transfer(address(bentoBox), amountToReturn);
}
```