Flash Loan

Blockchain Fundamentals Project Report

Konrad Wierzbik
konrad.wierzbik@gmail.com

Abstract

In this short report, I introduce the idea of a Flash Loan and discuss its implementation written in ink! 3.3 for smart contracts running on substrate blockchains.

1 Introduction

Flash Loan is a type of service/feature that is specific to decentralized finance (DeFi). It is a kind of uncollateralized loan that is given only for "the duration" of an atomic blockchain transaction. The loan is taken during the execution of some smart contract function and must be returned within the transaction or the whole transaction will fail and the loan will not be taken at all. There also exists, a similar type of service/feature called Flash Mint. In the case of Flash Mint a PSP22 Token (ERC20) allows the user to mint any amount of the token for the duration of one transaction.

The usages of Flash Loans in DeFi are to perform arbitrage operations and hack protocols. The former is that some smart contracts can perform arbitrage. A user wants to perform such an arbitrage operation but the user doesn't have the required funds to do so. What the user can do is, in a single transaction, take a flash loan, perform arbitrage, and finally return the flash loan. The latter is very similar. The only difference is that instead of performing an arbitrage operation, one uses a smart contract that can exploit other smart contracts and drain funds from them. This happens when authors of exploited smart contracts haven't considered that a single user can manage a huge amount of funds.

The important thing to have in mind during the implementation of the flash loan is to be aware of possible reentrancies and to make sure that at the end of the transaction, all the funds were returned. The implementation of an easily reusable and adjustable FlashLoan trait is described in what follows and can be found on GitHub [1].

2 Flash Loan implementation

The Flash Loan in short works as follows. A user or a contract calls a `flash_loan` in a contract that implements the **FlashLoan** trait. The contract sends the requested flash loan to a given accountId, and then calls the `execute_operation`, from **FlashLoanReceiver** trait, in the same contract. The receiver contract, if it implements **FlashLoanReceiver** trait, can manage the funds. At the end of `execute_operation`, the receiver should give allowance to the loan contract and then return with no error. Finally, at the end of `flash_loan` funds are transferred from the receiver back to the origin. Transferring from the receiver after the receiver `executes_operataion` assures that reentrancy attacks are ineffective.

2.1 Traits

FlashLoan trait

The FlashLoan Trait contains only one function that performs a flash loan - `flash_loan` with arguments:

- **receiver** - is an address of a smart contract that implements **FlashLoanReceiver** trait[1] that will receive the flash loan and may perform any operation using the loan before the loan is returned.

[1] It is needed so the contract giving flash loan can call a receiver so it can perform than any code that was implemented. Defined later.

- **assets**[] - is a list of assets that will be lent.
- **amounts**[] - is a list of amounts corresponding to the **assets**[] that will be lent.
- **data**[] - is a list of bytes that can be used for any purpose and it is passed to the receiver in default implementation. The data can be used by the receiver or by overriding the default implementation the data can be used inside the `flash_loan`.

FlashLoanReceiver trait

FlashLoanReceiver trait must be implemented by a contract that will receive a flash loan and then perform any operation. It contains one function that is called by a `flash_loan` function - `execute_operation` with arguments:

- **assets**[] - is a list of assets that were rent by a flash loaner.
- **amounts**[] - is a list of amounts corresponding to the **assets**[] that were rent by a flash loaner and will be taken back at the end of `flash_loan`.
- **fees**[] - is a list of fees corresponding to the **assets**[] that will be additionally taken from at the end of `flash_loan`.
- **data**[] - is a list of bytes that can be used for any purpose and was passed past by flash loaner.

2.2 Implementation

FlashLoan Impl

The implementation of FlashLoan comes to implementing `flash_loan`. The function implementation can be split into 6 steps separated into internal functions[2] that are called one after another:

- **_before_flash_loan** - by default does nothing. Can be overridden to perform any action, for example checking the initial conditions.
- **_calculate_fees** - by default returns a vector of fees filled with zeros. Can be overridden to calculate non-zero fees.
- **_send_flash_loan** - by default transfers requested amounts to the **receiver**, i.e. it gives the flash loan.
- **FlashLoanReceiver::execute_operation** - call the `execute_operation` in the **receiver**.
- **_get_back_flash_loan** - transfers from the **receiver** the given loan plus fees back to the lending contract.
- **_after_flash_loan** - by default does nothing. Can be overridden to perform any action, for example checking the final conditions.

[2]Internal functions start with underscore '_'.

All the above internal functions can be easily overridden to adjust the `flash_loan` implementation for any project needs. An example of such overridden implementation is presented in the *src/flash_loan_contract*[1].

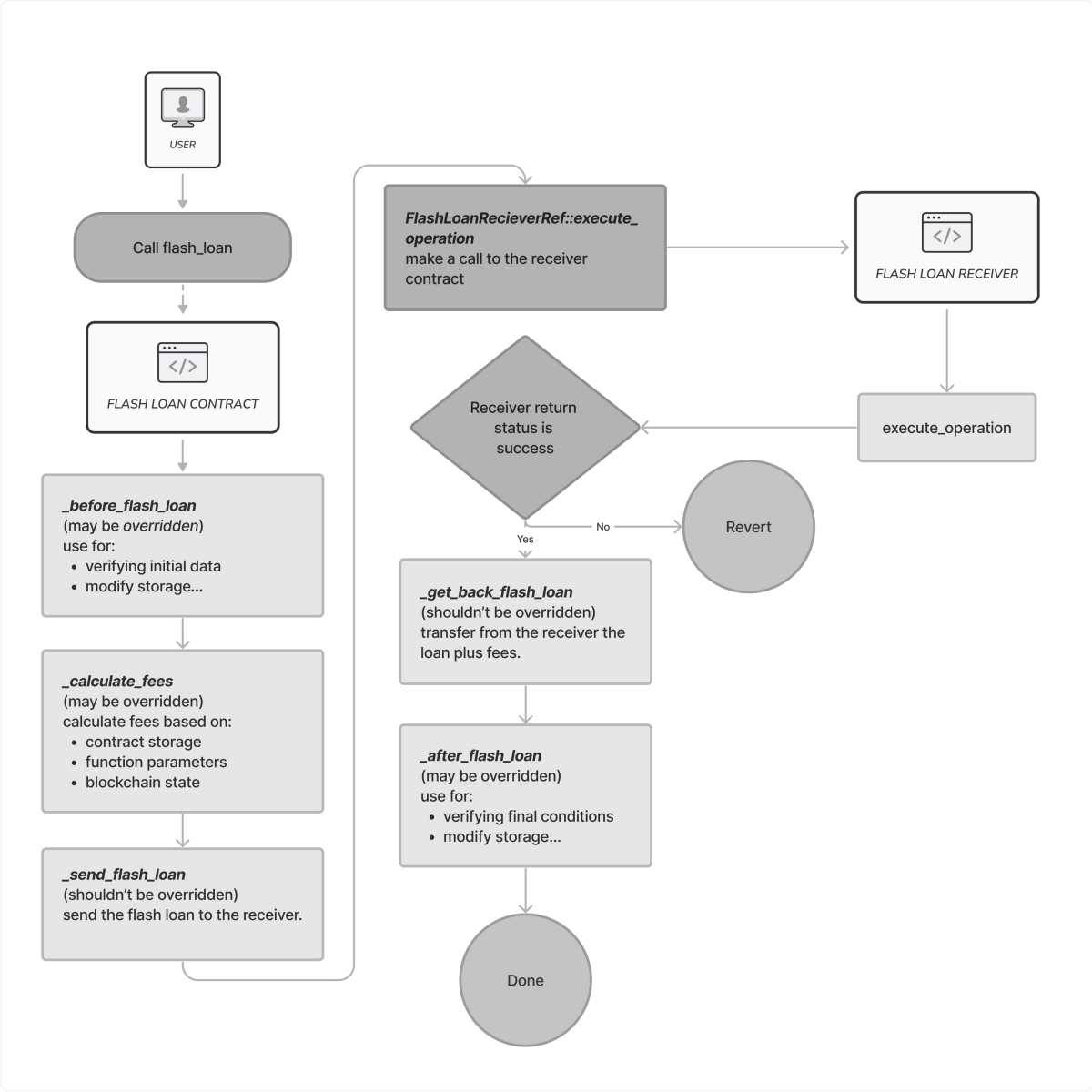

Figure 1: The `flash_loan` execution flow.

References

[1] https://github.com/DesiredDesire/flash_loan