Towards A First Step to Understand Flash Loan and Its Applications in DeFi Ecosystem

Dabao Wang[1], Siwei Wu[1], Ziling Lin[1], Lei Wu[1], Xingliang Yuan[2], Yajin Zhou[1], Haoyu Wang[3], and Kui Ren[1]

[1]Zhejiang University & Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province;
[2]Monash University; [3]Beijing University of Posts and Telecommunications;
{dabao.wang, wusw1020, linziling, lei_wu}@zju.edu.cn, xingliang.yuan@monash.edu,
yajin_zhou@zju.edu.cn, haoyuwang@bupt.edu.cn, kuiren@zju.edu.cn

ABSTRACT

Flash Loan, as an emerging service in the decentralized finance ecosystem, allows users to request a non-collateral loan. While providing convenience, it also enables attackers to launch malicious operations with a large amount of asset that they do not have. Though there exist spot media reports of attacks that leverage Flash Loan, there lacks a comprehensive understanding of existing Flash Loan services.

In this work, we take the first step to study the Flash Loan service provided by three popular platforms. Specifically, we first illustrate the interactions between Flash Loan providers and users. Then, we design three patterns to identify Flash Loan transactions. Based on the patterns, 76, 303 transactions are determined. The evaluation results show that the Flash Loan services get more popular over time. At last, we present four Flash Loan applications with real-world examples and propose two potential research directions.

KEYWORDS

Blockchain; DeFi; Flash Loan

ACM Reference Format:
Dabao Wang[1], Siwei Wu[1], Ziling Lin[1], Lei Wu[1], Xingliang Yuan[2], Yajin Zhou[1], Haoyu Wang[3], and Kui Ren[1]. 2021. Towards A First Step to Understand Flash Loan and Its Applications in DeFi Ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing (SBC '21), June 7, 2021, Virtual Event, Hong Kong.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3457977.3460301

1 INTRODUCTION

Decentralized finance, aka *DeFi*, has been growing in recent years. Up to 31th Jan 2021, the total value locked (TVL) [1] in *DeFi* has reached 28 billion USD [7]. A service called Flash Loan (i.e. uncollateralized loan), which does not exist in the traditional finance system, has drawn much attention. However, the introduction of Flash Loan is a double-edged sword.

On the one hand, it does bring in convenience [17] and facilitate the prosperity of *DeFi*. Traders without much capital can launch arbitrage, liquidation and asset swapping with Flash Loan. For instance, when traders discover a price difference among tokens between decentralized exchanges (DEXes), they can borrow a considerable amount of capital by Flash Loan (no collaterals are required) to maximize the profit.

On the other hand, Flash Loan also enables attackers to launch malicious operations with a large amount of capital that they do not have. Therefore, the attack consequences can be vastly amplified. In early 2020, two infamous incidents [16] caused a huge loss to bZx [4]. The hacker took the advantage of Flash Loan to manipulate the market price and made considerable profits of $0.83M$ USD and $1.1M$ USD, respectively. Most recently, a hacker borrowed $15M$ DAI [13] via Flash Loan to gain over $15M$ USD through repeatedly swapping tokens in the EMN [22] pool.

As such, there is an urgent need to demystify the Flash Loan ecosystem and understand the impact of potential security threats. Unfortunately, few studies have been proposed to serve this purpose. Specifically, previous studies mainly focused on profit optimization [6, 16] and oracles [12] used by protocols. To the best of our knowledge, none of them systematically demystified Flash Loan and its applications. To provide effective mitigations for Flash Loan, we still lack a comprehensive understanding of Flash Loan.

In this paper, we take the first step to systematically study Flash Loan and further design three patterns to identify Flash Loan transactions. Then, we determine all Flash Loan transactions until 31st Jan 2021 based on three proposed Flash Loan patterns. As a result, 76, 303 Flash Loan transactions are identified. To further demystify the behaviors behind Flash Loan, we perform an analysis on four Flash Loan applications and explain them with real-world examples.

This paper makes the following contributions:

- We study three Flash Loan providers to understand the working process of Flash Loan.
- We conduct full-chain measurements on Flash Loan transactions launched in the *DeFi* ecosystem. As far as we know, this is the first work to give a measurement for Flash Loan transactions based on real-world data.
- We describe four types of Flash Loan applications with real-world examples and propose two potential research directions.

[1] TVL of a specific protocol represents the total amount of assets staked by users.

SBC '21, June 7, 2021, Virtual Event, Hong Kong
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8405-6/21/06…$15.00
https://doi.org/10.1145/3457977.3460301

Paper organization: The rest of the paper is arranged as follows. Section 2 presents the related work. Section 3 elaborates on some concepts in Ethereum and *DeFi*. Section 4 illustrates the general idea of Flash Loan and explains its working process under different platforms. Section 5 conducts an evaluation on Flash Loan transactions. Section 6 describes four applications behind Flash Loan. Section 7 proposes two potential research directions. Finally, we conclude the paper in Section 8.

2 RELATED WORK

Kaihua et al. [16] investigate two existing exploits that happened on 15th and 18th Feb 2020 and present the details of how traders leverage the Flash Loan mechanism with the trick of price manipulation to gain profits. They also propose a process to re-boost two exploits via optimized parameters. Lewis et al. [10] leverage Flash Loan to execute a governance attack [23] on MakerDAO [14]. Moreover, the proposed strategy leads to a theft of 0.5B USD and unlimited mining of DAIs. Bowen et al. [12] systematically study 4 oracle designs in *DeFi* via comparing their price deviations. Besides, they exhibit the potential vulnerabilities existing among 4 oracle designs. Kamps et al. [11] aggregate the information from the existing pump-and-dump schemes among the classic economic and propose a group of patterns with summarised criteria to identify potential pump-and-dump activities in crypto markets. Xu et al. [21] also investigate 412 pump-and-dump activities to build a model that predicts the pump behavior for all assets exhibiting in DEXes by estimating its pump likelihood. Philip et al. [6] present the breadth of arbitrage bots and their profit-making strategies, which optimize users' network latency and pay a high transaction gas fee to win priority gas auctions (PGAs). Furthermore, they highlight that bots' revenue far exceeds the Ethereum block reward and transaction fees. They state that the blockchain consensus stability might be threatened with such high optimization fees. Eskandari et al. [9] also study the front-running issues across the 25 most active decentralized applications (DApps) on the Ethereum blockchain and summarized their proposed solutions into useful categories.

3 BACKGROUND

The introduction of the blockchain technique [15] has changed the financial ecosystem in the world. Especially with the invention of Ethereum [20], there has been a wave of developing the decentralized applications (DApps). Smart contracts, as the basis of DApps, enable a transparent environment and become essential components for the development of *DeFi*.

3.1 Common concepts on Ethereum

To make this work easy to understand, We first introduce a few common concepts in Ethereum.

Account. Ethereum is an account-centric blockchain system. There are two types of accounts: External Owned Account(EOA) and smart contract account (smart contract in short). The main difference between them is that EOAs are controlled by private keys, and smart contracts are controlled by codes. Basically, an EOA is created with the generation of the public and private key pair, and a smart contract is always created by an EOA or another smart contract. Both EOAs and smart contracts are identified by their addresses, like `0x16431837a35b5469675b2ba5d9b7575d25b721c3`.

Digital Currency. Ether and the ERC20 token are two main types of digital currencies in Ethereum. Compared to Ether which is supported natively, ERC20 tokens are supported by smart contracts. Once a smart contract implements the interfaces of ERC20 token standard [1], then the smart contract can act as an ERC20 token. Moreover, ERC20 tokens can only be transferred by invoking two ERC20 token standard functions: *transfer* and *transferFrom*.

Transaction. All actions on the Ethereum blockchain are based on transactions. A transaction have three purposes: transferring Ether, invoking a smart contract function, and deploying a smart contract. According to the circumstance, there exist two types of transactions: *external* transaction and *internal* transaction. The *external* transactions are initiated from EOAs. Alternatively, once a smart contract is invoked within an *external* transaction, the *internal* transaction will be triggered. The word "transaction" written individually in the remaining paper indicates the collection of an *external* transaction.

Gas Fee. Gas is the unit to measure the computational resource used to run operations in Ethereum. To execute transactions on the Ethereum Network requires users to pay a certain amount of Ether (known as gas fee). The gas fee is equal to the value that gas used in the transaction times provided gas price. Since there is no limitation on defining the gas price, users can control the gas fee for their transactions by setting any gas price. Typically, the higher the gas price is, the faster the transaction is verified on the Ethereum blockchain.

Function and Event. The smart contract function is identified by the function signature, which is the first four bytes of the hash value (SHA3) of the function name with the parenthesized list of parameter types. If a user sets a function signature in front of a transaction's call data, then the callee smart contract's corresponding function will be invoked. Smart contracts' developers usually leverage the event to record critical information. For example, the ERC20 token standard specifies an event *Transfer* to record the spender, receiver and amount of transferred ERC20 tokens. Similarly, an event is identified by the hash value (SHA3) of the event name with the parenthesized list of parameter types. When an event is triggered, a log with an event hash is recorded in Ethereum.

3.2 Primitives in *DeFi*

Decentralized finance is a transparent and permissionless finance ecosystem without relying on intermediaries such as banks. In Ethereum, *DeFi* is formed with open-source protocols deployed as smart contracts. In the following, we will introduce some primitives in *DeFi*.

Decentralized Exchange (DEX). In the centralized exchanges (CEXes), users entrust their capital to CEXes for trading, and CEXes need to guarantee security. Conversely, trading on DEXes does not require users to provide access to their private keys. Therefore, users can still have full control of their capital.

In particular, there are two main types of DEXes: *Order Book* and *Automated Market Maker* (AMM). *Order Book* DEXes usually maintain a list of buy and sell orders. They match the pair of orders with a compatible price in their database. As for AMM DEXes, they

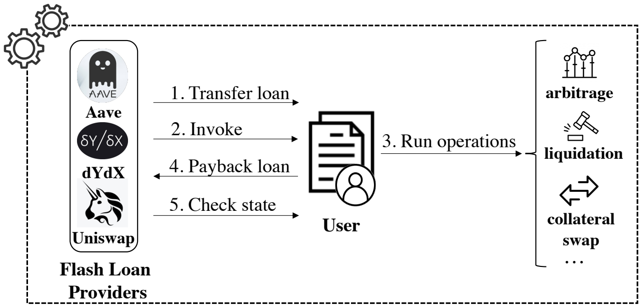

Figure 1: The workflow of a Flash Loan transaction.

maintain various liquidity pools with a designed price calculating mechanism. The price of assets lying in the pool is usually calculated based on its price mechanism and existing liquidity. In comparison, trading in AMM DEXes is more flexible because there is no need for the matching process.

Lending. Lending platforms share interests for depositors to lock their capital in the liquidity pool and provide a collateral loan for borrowers. The lending platforms normally require traders to deposit more collateral than the borrowed assets with a certain ratio. Most of the lending platforms design a protection mechanism called liquidation to prevent the potential loss caused by price slippage on traders' deposited collaterals. Once the collateralization ratio (*collateral value*/*debt value*) is reached, the lending platforms will first sell lenders' collateral with a discount to liquidators. Then, a certain percentage of collateral will be charged to lenders as a penalty. We will discuss more details about liquidation in Section 6.

4 FLASH LOAN

In this section, we first explain the general idea of Flash Loan. Second, we elaborate on three famous Flash Loan providers and compare their differences in requirements of using. Furthermore, we summarize a Flash Loan pattern for each provider to identify Flash Loan transactions.

4.1 General Idea of Flash Loan

To request a loan in *DeFi* platforms, the user is usually required to deposit overcollateralized assets (i.e., digital cash or tokens). However, a new functionality called Flash Loan is developed to enable a non-collateral borrowing service. Moreover, a considerable amount of assets can be "generously" lent to users by Flash Loan as long as the borrowed assets can be paid back within the current transaction. Otherwise, the platform will instantly revert the transaction to get the lent assets back.

Figure 1 presents the general workflow of a Flash Loan transaction. There are two main entities: *Flash Loan Providers* and *Users*. To interact with *Flash Loan Providers*, *Users* are required to develop a smart contract. A user's contract usually includes three parts: 1) borrowing the loan(s) from Flash Loan providers, 2) interacting with other smart contracts, and 3) returning the loan(s). To our best knowledge, there are three main Flash Loan providers [2] [8] [19] supporting the Flash Loan service with or without certain fees.

Specifically, we generalize the workflow of a Flash Loan transaction into five steps. First, *Flash Loan Providers* transfer requested assets to *Users*. Second, they invoke *Users*' pre-designed operations. Third, *Users* will interact with other contracts to execute operations with borrowed assets. Once the execution is completed, *Users* have to return the borrowed assets with or without the extra fee charged by *Flash Loan Providers*. Finally, *Flash Loan Providers* will check their balance. If they discover that no or non-sufficient assets are returned by *Users*, they will revert the transaction immediately. Note that all five steps are finished in one transaction.

4.2 Flash Loan Providers

In this section, we give a basic introduction of each Flash Loan provider and reveal their fee-charging mechanisms. Besides, we explain how users' smart contracts interact with different Flash Loan providers.

4.2.1 Aave.

Aave [2] is currently the second largest lending platform that locks over $\$3.72B$ [7] up to Jan 2021. As the first platform officially providing the Flash Loan service, *Aave* provides a native function called `flashLoan` [2] designed in *Aave*'s official contract, i.e., the `LendingPool` contract, to trigger Flash Loan. Moreover, requesting Flash Loan in *Aave* charges 0.25% of the borrowed assets as the fee.

How to prepare flash loan contract with Aave. For *Aave*'s Flash Loan, users need to develop a smart contract consisting of one execution function and one entry-point function. The execution function contains users' designed operations for the loaned assets, e.g., trading in exchanges. Note that, the execution function has to be formed based on `executeOperation` [3] designed in *Aave*'s official contract `FlashLoanReceiverBase`.

In the entry-point function, users first need to prepare the function `flashLoan` to request a loan. Second, users can follow up with the function `executeOperation` to run the designed logic on the loaned assets. Third, returning loaned assets must be completed with the provided function `transferFundsBackToPoolInternal` [4] after finishing executing operations. If *Aave* discover that the vault's state is not balanced (no or non-sufficient assets are paid back to the vault), it will instantly revert the entire transaction. Once the preparation for the contract is done, users can deploy their contract to the chain and use the Flash Loan service from *Aave* by invoking the entry-point function.

Identify flash loan transactions from Aave. As aforementioned, *Aave* exposes a native function called `flashLoan` for users to utilize *Aave*'s Flash Loan. Once the function `flashLoan` is invoked successfully, it emits a unique event called `FlashLoan` [5]. Therefore, we can use this feature to identify Flash Loan transactions from *Aave*. As a result, we discover that there exist over $15,000$ transactions including *Aave*'s Flash Loan up to 31st Jan 2021.

4.2.2 dYdX.

dYdX [8] is a non-custodial platform providing services mainly including lending and borrowing on their supporting cryptoassets like ETH, USDC, and DAI. At the time of writing this

[2] function flashLoan(address _receiver, address _reserve, uint256 _amount, bytes calldata _params)
[3] executeOperation(address _reserve, uint256 _amount, uint256 _fee, bytes calldata _params)
[4] transferFundsBackToPoolInternal(address _reserve, uint256 _totalDebt)
[5] FlashLoan(address indexed _target, address indexed _reserve, uint256 _amount, uint256 _totalFee, uint256 _protocolFee, uint256 _timestamp)

paper, *dYdX* has locked over $157M$ USD. According to our investigation, there is no native Flash Loan feature provided by *dYdX*. However, *dYdX*'s `SoloMargin` contract provides a function called `operate` [6] that enables to bring a series of operations into one transaction to achieve Flash Loan for users. Surprisingly, *dYdX* does not charge any fee for invoking the function `operate`.

How to prepare flash loan contract with dYdX. The preparation for Flash Loan in *dYdX* is pretty similar to *Aave*'s. Users are required to develop a contract including one execution function, which contains users' operating logic on the loaned assets, and one entry-point function. In the entry-point function, users first need to sequentially organize a list of provided (by *dYdX*) actions: `withdraw`, `callFunction` [7] and `deposit`. Then, users can leverage the function `operate` to run the actions one by one to perform Flash Loan logic. Note that, `callFunction` is acting as the execution function mentioned above to perform users' operating logic. In details, `withdraw` helps users borrow assets from *dYdX* without any collateral. Then, `callFunction` is executed to run users' particular operations on the loaned assets. Finally, *deposit* pays back the loan. Once the contract is well prepared and deployed on the chain, users can run Flash Loan in *dYdX* by invoking the entry-point function.

Identify flash loan transactions from dYdX. Though *dYdX* does not directly provide Flash Loan feature, users can still achieve Flash Loan service in *dYdX* by sequentially executing a series of actions: *Operate*, *Withdraw*, *callFunction*, *Deposit.* Note that, all actions have corresponding event logs: `LogOperate`, `LogWithdraw`, `LogCall`, and `LogDeposit`. Therefore, to identify transactions containing *dYdX*'s Flash Loan service, two conditions should be checked. First, all actions' event logs should exist in a transaction. Second, all event logs have to follow a particular order showed below:

`LogOperate` ⟶ `LogWithdraw` ⟶ `LogCall` ⟶ `LogDeposit`

Once two conditions are both satisfied in a transaction, we can confirm that it is a Flash Loan transaction from *dYdX*. Based on our experiment, around $25,000$ transactions are identified as Flash Loan transactions leveraging *dYdX*'s service.

4.2.3 UniswapV2. As one of the most famous DEX protocols, *Uniswap* [19] occupies around 11% ($3.2B$ USD) of liquidity in *DeFi* ecosystem. Different from *Aave* and *dYdX*, *Uniswap* simply builds its Flash Loan feature called *flash swap* on the function `swap` [8]. In details, the function `swap` switches in between flash swapping and normal swapping based on the provided parameters. In terms of the fee, compared to the aforementioned Flash Loan providers, *UniswapV2* charges the highest fee (0.3%) based on users' borrowed assets.

How to prepare flash loan contract with UniswapV2. A contract for using Flash Loan in *UniswapV2* requires users code their designed operations in function `IUniswapV2Callee` which inherits from *UniswapV2*'s interface standard `IUniswapV2Callee` . The designed operations must include repayment action to success *flash swap.* Unlike the two Flash Loan providers previously analyzed, users do not need to develop any entry-point function to initiate a transaction. Instead, users first neet to find the targeted pair contract [9] published by *Uniswap.* Then, through invoking the function `swap` with specific parameters, a Flash Loan transaction will be triggered. In particular, parameter `to` should be the address of the deployed contract, and the length of parameter `data` should be greater than zero.

Table 1: The distribution of Flash Loan transactions.

Providers	# Transactions	# Receivers	# Average Use
Aave	$15,016$	463	32.4
dYdX	$24,983$	666	37.5
UniswapV2	$36,574$	346	105.7

Identify flash loan transactions from UniswapV2. Identifying Flash Loan transactions from *UniswapV2* requires three steps. First, we verify the event `PairCreated` emitted by the *UniswapV2Factory* contract and collect a group of pair contracts (addresses) that supplies the `swap` function. Second, we verify the event `swap` emitted by triggering the function `swap` in all transactions. Lastly, once we confirm that the transaction invokes the `swap` function of pair contracts, we identify it as Flash Loan transaction from *UniswapV2* through checking three conditions:

(1) The length of the parameter *data* is greater than zero.
(2) The internal transaction triggered by `uniswapV2Call` must include the invocation of `transfer` or `transferFrom` function.
(3) The receiver address of `transfer` or `transferFrom` function must be the pair contract.

In conclusion, if all three conditions are fulfilled, the transaction can be confirmed as a operation of *flash swap* (i.e., Flash Loan in *UniswapV2*). Through applying the identifying pattern, around $36,500$ transactions are filtered.

Importantly, since our patterns strongly rely on the specific rule published by each Flash Loan provider, the identified Flash Loan transactions will not result in any false positive.

5 THE MEASUREMENT OF FLASH LOAN TRANSACTIONS

In this section, we perform a measurement for Flash Loan transactions. As for the data set, we collect about one billion transactions (up to 31th Jan 2021) from the Ethereum blockchain ledger.

Statistic Result. Through applying three Flash Loan patterns proposed in Section 4 over one billion collected transactions, $76,303$ Flash Loan transactions are identified. As shown in Table 1, users leverage Flash Loan in *UniswapV2* most frequently, as there are $36,574$ (48%) transactions. Following up, $24,983$ (32.5%) Flash Loan transactions are found in *dYdX*, and $15,016$ (19.5%) Flash Loan transactions are found in *Aave*. The second last column of Table 1 records the amount of Flash Loan receivers for each Flash Loan provider. According to the results, Flash Loan in *dYdX* has been used by most unique receivers (666) as well as there are 463 and 346 Flash Loan receivers in *Aave* and *UniswapV2*, respectively.

[6] function operate(Account.Info[] memory accounts, Actions.ActionArgs[] memory actions)

[7] callFunction(address sender, Account.Info memory account, bytes memory data)

[8] swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)

[9] Uniswap has many pair contracts, which are published for users, to launch a swap on a pair of tokens. In *UniswapV2*, every pair contract supplies `swap` function.

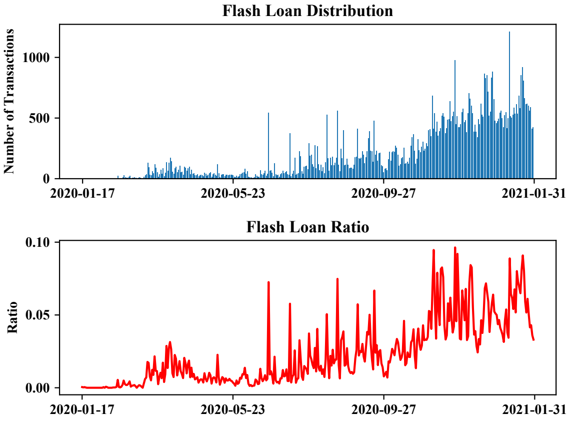

Figure 2: The distribution of Flash Loan transactions.

In the last column of Table 1, it presents the average number of transactions launched by each receiver.

Trend. To further understand the popularity of Flash Loan in *DeFi*, we provide two time series for the number and the ratio (the number of Flash Loan transactions / the number of all transactions) of Flash Loan transactions in Figure 2. Note that, both the number and the ratio of Flash Loan transactions are increasing over the time.

Finding. Nowadays, Flash Loan service is getting more popular. Through measuring Flash Loan transactions, we discover that Flash Loan in *UniswapV2* is used most intensively (105.7 transactions per receiver), while Flash Loan in *Aave* (32.4 transactions per receiver) is used least intensively.

6 APPLICATIONS OF FLASH LOAN

Flash Loan can be used for legitimate purposes such as arbitrage, liquidation, etc. Besides, Flash Loan can also become a sharp knife for aggressive users to harm the *DeFi* ecosystem. In this section, we describe four applications of Flash Loan and discuss the benefit that Flash Loan brings to them.

6.1 Arbitrage

General speaking, arbitrage in *DeFi* is a behavior to gain benefits by trading in between platforms supplying different price for an asset. Since the *DeFi* market reacts slower for events happening in the network than the real-world market, traders can take advantage of the market's inefficiencies to buy and sell the cryptoassets at a different price to gain financial benefits. Note that, arbitrage itself is not a malicious behavior. In fact, the arbitrage can be leveraged to balance the token prices between DEXes.

Benefit. With Flash Loan, traders can launch arbitrage without any pre-owned asset. In particular, if the price difference is found, the arbitrageurs can instantly borrow a considerable asset with Flash Loan service to earn benefits. Therefore, arbitrages with Flash Loan become "cost-free" as long as traders can afford the gas fee to launch the transaction.

Example. We will present an arbitrage that happened on 22nd Jan 2021 with four steps. First, in the transaction [10], the trader borrowed 1.13 Ether from *dYdX*. Second, 1.13 Ether was converted to 345 LPT tokens in *Balancer* [3] through a trade. Third, another trade was triggered to trade 345 LPT tokens on the corresponding liquidity pool of *Uniswap*. As a result, 1.46 Ether were gained by the trader. Finally, the trader returned 1.13 Ether to *dYdX* from gaining. In this arbitrage case, the trader gained 0.33 Ether (around 538 USD at the time) by paying 0.05 Ether as the gas fee.

6.2 Wash Trading

Wash trading in *DeFi* is a behavior that creates fake trading volume for certain cryptoassets or platforms. Specifically, wash trading is a group of trades increasing the trading volume on the asset or platforms. In reality, wash trading can easily mislead users to perform financial operations on the targeted cryptoassets and platforms. Though some countries like the U.S. have banned washing trading to protect their traditional markets and the stock market, it is brought back to the crypto market again because of the popularity of cryptocurrency and the lack of legal management.

Benefit. With Flash Loan, wash traders can manipulate the market without a large amount of capital as long as they can afford the potential loss and the gas fee.

Example. In the transaction [11] executed on 14th Jul 2020, the trader firstly borrowed 10 Ether from *Aave*. Then, five repeated trades were launched to increase the trading volume in the liquidity pool *"Uniswap V2: DAI 2"*. In details, for each trade, 2 Ether was first converted to *DAI* and all *DAI* would be instantly converted back to Ether at the pool. Finally, the user paid back the Flash Loan. There were no further operations except wash trading in this transaction. As a result, the trader lost 0.068 Ether in this transaction and paid 0.164 Ether as the gas fee.

6.3 Flash Liquidation

Liquidation is a behavior launched by the liquidator to buy undercollateralized assets from the lending platforms. There are two liquidation classes (*Fixed Price Biding* and *Auction*) involving three roles (platforms, liquidators and collateral keepers). For fixed price biding, the lending platforms like *dYdX* and *Compound* allow liquidators to buy undercollateralized assets from collateral keepers with a specific discount. Moreover, the lending platforms will apply a fixed amount of liquidation penalty to collateral keepers. Alternatively, the platforms like MakerDAO [14] allow liquidators to compete on the keeper's undercollateralized assets like an auction. The winners, who pay the higher gas fee to launch their transactions, can buy the undercollateralized collateral with a discount.

Benefit. With Flash Loan, anyone can become a liquidator to make profits without much capital by buying the undercollateralized assets with a specific discount.

Example. The transaction [12] performed a liquidation on 3rd Nov 2020. In details, the liquidator first borrowed $12,940$ *DAI* from *dYdX* and swapped the *DAI* to $13,046$ *USDT* [18]. Second, $13,046$ *USDT* was used to buy the asset from the undercollateralized position in

[10] 0x2a0c2599f89d95a46f4f28712e99a71847d8f72af5bdc8942d6c9dd01d896624
[11] 0x8fc77fa516aca91715046c1f307397ac49d211244fced5734c480a660015f927
[12] 0x38b706beda9426027081f2b6c1f2d6e68b2387d824a59e20a4d6decdfec43385

Compound [5]. Through exchanging the bought asset, the liquidator got 13,450 *DAI*. Finally, after paying back the Flash Loan, 510 *DAI* (about 510 USD) remained as profits, which is greater than the gas fee (about 172 USD).

6.4 Collateral Swap

Collateral swap in *DeFi* is a well-defined behavior consisting of two main steps:

- *Swapping:* Redeeming the collateral from the old loan.
- *Operating:* Lauching operations on redeemed collateral.

Since the crypto market is extremely unpredicted, timely closing existed collateral position becomes an urgent need for the holder to stop loss from severe slippages and liquidations.

Benefit. For users without sufficient capitals for *Swapping*, Flash Loan can solve their urgent need by providing "cost-free" assets to save their collaterals from the price slippage and the liquidation. Besides, Flash Loan also enables *Swapping* and *Operating* actions run within one transactions. It further prevents users from suffering the uncertainty (like slippage) between transactions.

Example. In the transaction [13] launched on 17th Mar 2020, the user performed a collateral swap to take out *BAT* (*Swapping*) and exchanged it to more stable asset *USDC* (*Operating*). As the context, the user previously opened a loan of *DAI* by depositing the asset *BAT*. In details, first, the user borrowed 25 *DAI* Flash Loan from *Aave*. Second, 25 *DAI* was paid to redeem out 504 *BAT* (collateral). Third, the redeemed collateral was further swapped to a more stable asset *USDC*. Lastly, the user converted part of the *USDC* to pay back *Aave*'s Flash Loan. As a result, the user swapped his/her collateral to a more stable asset without holding any *DAI*.

In conclusion, Flash Loan provides a vast convenience for multiple applications (arbitrage, wash trading, liquidation, and collateral swap) in *DeFi* ecosystem. It can either bring traders benefits or be used maliciously. Thus, understanding the intention of the application is necessary for both traders and developers.

7 FUTURE RESEARCH DIRECTIONS

In this section, we propose two potential research directions.

Arbitrage. Arbitrages in *DeFi* happen nearly every day. By leveraging the smart contract and Flash Loan, many organizations and individuals create bots to launch designed operations. We believe that the arbitrage bots in *DeFi* can maximize traders' profits if the information (i.e. price difference) can be timely detected and fed to the bot.

DeFi Attacks. With the increasing popularity of *DeFi*, attackers could steal money from *DeFi* platforms or individuals. Identifying malicious transactions, especially the zero-day attacks, is challenging due to complicated interactions between multiple entities (Figure 1). How to propose effective methodologies to detect attacks towards DeFi platform is still an open research question.

8 CONCLUSION

This paper takes the first step to study the working process of Flash Loan within three different platforms. In this work, we identified 76,303 Flash Loan transactions and 1,454 Flash Loan receivers. Furthermore, we evaluated the popularity of Flash Loan. To better understand the application behind the Flash Loan mechanism, we elaborated on four types of applications with real-world examples. Finally, we proposed two potential research directions in this area.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for the thorough reviews. This work was supported by the Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang (2018R01005), and the Fundamental Research Funds for the Central Universities (K20210226).

REFERENCES

[1] 2015. ERC20 Token Standard. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[2] Aave. Last Accessed: Sep. 2020. Aave. aave.com.

[3] Balancer. Last Accessed: Oct. 2020. Balancer Exchange. balancer.exchange.

[4] bZx. Last Accessed: Sep. 2020. bZx. bzx.network.

[5] Compound. Last Accessed: Oct. 2020. Compound. https://compound.finance/.

[6] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2019. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234* (2019).

[7] defipulse. Last Accessed: Sep. 2020. DEFI PULSE. https://www.defipulse.com/.

[8] dYdX. Last Accessed: Sep. 2020. dYdX. dydx.exchange.

[9] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2019. Sok: Transparent dishonesty: front-running attacks on blockchain. In *International Conference on Financial Cryptography and Data Security*. Springer, 170–189.

[10] Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshits. 2020. The Decentralized Financial Crisis: Attacking DeFi. *arXiv preprint arXiv:2002.08099* (2020).

[11] Josh Kamps and Bennett Kleinberg. 2018. To the moon: defining and detecting cryptocurrency pump-and-dumps. *Crime Science* 7, 1 (2018), 18.

[12] Bowen Liu and Pawel Szalachowski. 2020. A First Look into DeFi Oracles. *arXiv preprint arXiv:2005.04377* (2020).

[13] Maker. Last Accessed: Sep. 2020. DAI Token. etherscan.io/token/0x6b175474e89094c44da98b954eedeac495271d0f.

[14] MakerDAO. Last Accessed: Oct. 2020. MakerDAO. makerdao.com.

[15] Satoshi Nakamoto et al. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[16] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. 2020. Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. *arXiv preprint arXiv:2003.03810* (2020).

[17] DeFi Saver. Last Accessed: Oct. 2020. DeFi Saver. defisaver.com.

[18] Tether. Last Accessed: Jan. 2021. Tether: Fiat currencies on the Bitcoin blockchain. https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf.

[19] Uniswap. Last Accessed: Sep. 2020. Uniswap. uniswap.org.

[20] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.

[21] Jiahua Xu and Benjamin Livshits. 2019. The anatomy of a cryptocurrency pump-and-dump scheme. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1609–1625.

[22] Yearn.finance (YFI). Last Accessed: Sep. 2020. EMN Token. etherscan.io/token/0x8b31d4A56dD29d18C817ef0fA3990d30ECC5D89b.

[23] Micah Zoltu. 2020. How to turn $20M into $340M in 15 seconds. https://medium.com/coinmonks/how-to-turn-20m-into-340m-in-15-seconds-48d161a42311.

[13] 0xaf4ca18a0d3b94d948a9eeb47ba57c84c212aaeb7284b38ede6a0f6f549c3827