

Master Thesis

# Flash Loans and Decentralized Lending Protocols: An In-Depth Analysis

Florian Gronde 2012-065-694

Supervised by:

Prof. Dr. Fabian Schär

Credit Suisse Asset Management (Schweiz) Professor for

Distributed Ledger Technologies and Fintech

Center for Innovative Finance, University of Basel

## **Abstract**

This paper deals with the concept of Flash Loans on the Ethereum Blockchain. It presents the unique concept and elaborates on different applications. It therefore shows the functionality on the platforms Aave, dYdX and bXz. Furthermore, it deals with comparable concepts like Flash Swaps on Uniswap and Flash minting. It also describes the two largest flashloan attacks that have brought the concept into the public eye. For an economic analysis, Flash Loans on the described platforms were analyzed and their applications were worked out. It is shown how the usage is distributed to the individual platforms and applications and which tokens are in focus. It is pointed out that the most common applications are arbitrage in various forms and closing of collateral debt positions. Further it is shown that the platform dYdX is mainly used for arbitrage and Aave for closing of collateral debt positions, whereas Uniswap and flash minting is used less.

**Keywords:** Aave, bZx, Flash loans, Flash minting, Flash swaps, Uniswap, uncollateralized loans.

# Contents

<b>List of figures</b>	<b>II</b>
<b>List of tables</b>	<b>III</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Flash Loans - an introduction . . . . .	2
1.2 Relevance of the topic . . . . .	3
1.3 Objective . . . . .	4
1.4 Methodology . . . . .	4
1.5 Structure of the thesis . . . . .	5
<b>2 Decentralized loans</b>	<b>6</b>
2.1 Classic P2P lending . . . . .	6
2.2 Flash loans . . . . .	8
2.2.1 Applications . . . . .	9
<b>3 Flash loans in detail</b>	<b>15</b>
3.1 Aave Lending Protocol . . . . .	15
3.2 dYdX . . . . .	16
3.3 bZx . . . . .	17
3.4 Comparable approaches . . . . .	18
3.4.1 FlashSwaps . . . . .	18
3.4.2 Flash-minting . . . . .	21
3.5 Flash loan attacks . . . . .	22
<b>4 Empirical research</b>	<b>26</b>
4.1 Data acquisition and preparation . . . . .	26
4.2 Categorization . . . . .	29
4.3 Findings . . . . .	36
4.3.1 Utilization numbers of Flash Loans . . . . .	37
4.3.2 Distribution of txns over time . . . . .	39
4.3.3 Applications . . . . .	41
4.4 Token on protocols . . . . .	50
4.5 USD volumes . . . . .	53
<b>5 Conclusion</b>	<b>59</b>
5.1 Limitations and further research . . . . .	60
<b>References</b>	<b>i</b>
<b>Appendix</b>	

## List of Figures

1	Token transfers of first flashloan . . . . .	9
2	Token transfers of CCDP example . . . . .	11
3	Token transfers of collateral swap example . . . . .	13
4	Token transfers of wash trading example . . . . .	14
5	Distribution of flashloan txns over protocols . . . . .	39
6	Distribution of txns on protocols over time . . . . .	41
7	Distribution of txns over applications . . . . .	43
8	Distribution of txns over applications subcategories . . . . .	46
9	Distribution of applications on all protocols over time . . . . .	47
10	Linear regression CCDP txns and price of ETH . . . . .	48
11	Distribution of arbitrage bot txns over time on protocols . . . . .	49
12	Flash Loan value in mio USD per month . . . . .	55
13	USD volume of Flash Loans on protocols per month . . . . .	56
14	Volume of Flash Loans for applications in USD . . . . .	57
15	Volume of Flash Loans for applications subcategories in mio USD . . . . .	58

## List of Tables

1	Attributes in final data set . . . . .	36
2	Distribution of Flash Loans over protocols . . . . .	38
3	monthly usage of Flash Loans . . . . .	40
4	txn count over time . . . . .	41
5	Distribution of applications over flashloan txns . . . . .	42
6	Distribution of subcategories over protocols . . . . .	44
7	Txns error of failed txns on Uniswap . . . . .	45
8	Top three CCDP txns per day . . . . .	47
9	Top five potential hammer bot txns per day . . . . .	50
10	Flash Loan amounts and distribution of tokens on Aave .	51
11	Flash Loan amounts and distribution of tokens on dYdX	51
12	Flash Loan amounts and distribution of tokens on Uniswap	52
13	Flash Loan value in mio USD per month . . . . .	54
14	Flash Loan value in USD per protocol . . . . .	56

## List of abbreviations

**API** Application Programming Interface

**BAT** Basic Attention token

**CDP** Collateralized Debt Position

**dApps** Decentralized Applications

**DeFi** Decentralized Finance

**DEX** Decentralized Exchange

**ETH** Ether

**EVM** Ethereum Virtual Machine

**P2P** Peer-to-Peer

**SAI** Single-Collateral DAI

**SC** Smart Contract

**txn** transaction

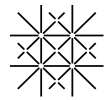
**USD** US Dollar

**USDC** US Dollar Coin

**WBTC** Wrapped Bitcoin

**WETH** Wrapped Ether

**XRP** Ripple



University  
of Basel

Center for  
Innovative Finance

## Plagiatserklärung

Ich bezeuge mit meiner Unterschrift, dass meine Angaben über die bei der Abfassung meiner Arbeit benutzten Hilfsmittel sowie über die mir zuteil gewordene Hilfe in jeder Hinsicht der Wahrheit entsprechen und vollständig sind. Ich habe das Merkblatt zu Plagiat und Betrug vom 22. Februar 2011 gelesen und bin mir der Konsequenzen eines solchen Handelns bewusst.

Florian Gronde

2012-065-694

## Acknowledgements

Many thanks to Prof. Dr. Fabian Schär for the opportunity to work on such an exciting topic. Thanks also to the developer community of dYdX and to Noah Zinsmeister of Uniswap, Antonio Juliano of bZx and Austin Williams of the project flash minting for tips on flashloan identification and considerations. Huge thanks to Susanne Halimi for all the technical explanations and thoughts during the data acquisition and the push to dive deeper into the topic by the overwhelming interest in my work and challenging discussions. It was a joy to bring all the complex thoughts together and learn from each other.



# 1 Introduction

## 1.1 Flash Loans - an introduction

On 15 February 2020, a transaction could be observed on the Ethereum Blockchain<sup>1</sup> that was unique at the time. Within under one minute, one block and in one transaction, a profit of about US Dollar (USD) 360,000 was achieved through the advantageous use of a smart contract. The news went viral quickly (e.g. Trustnodes (2020*b*)) and drew public attention to the transactions involved. The gain could be achieved by a nearly risk-free loan in the form of a flashloan and the subsequent exploitation of arbitrage between different decentralized cryptocurrency exchanges. First, a flashloan of 10,000 Ether (ETH) from the dYdX trading platform was invested half on the Compound platform for conversion into Wrapped Bitcoin (WBTC) and half on the bZx platform for shortening the same amount WBTC. Subsequently, the price of WBTC was lowered by means of a swap on the Uniswap platform, and the shortening resulted in a profit from the settlement. The flashloan and the protocol fee were repaid and a profit of approximately USD 360,000 was achieved. Flash loans are currently offered by various platforms. The most prominent is probably Aave and dYdX. Aave developers (2020*b*) describe Flash Loans as a possibility “to borrow in an undercollateralized fashion from the reserve [...] a certain amount of liquidity, that must be returned within the same transaction.” Besides a relatively low protocol fee, there are no further costs for the borrower. The only condition is that the loan must be returned within the same transaction, within one block on the Ethereum Blockchain. Although the very short time between taking out a loan and repaying it seems strange at first, the previous example impressively shows the possibilities of this relatively young concept. Although the possibility of extreme capital gains by manipulating the price of other currencies, as exploited in February and only possible because of a bug, has now been eliminated for the described procedure, the basic concept

---

<sup>1</sup>transaction (txn):

0xb5c8bd9430b6cc87a0e2fe110ece6bf527fa4f170a4bc8cd032f768fc5219838

of risk-free loans nevertheless opens up a wide range of investment opportunities. It enables uncollateralized loans. That is, loans that do not require collateral that exceeds the value of the loan. Collateralized loans are probably the most common form of lending in the traditional financial sector and on most Decentralized Finance (DeFi) trading platforms for cryptocurrencies. Before a loan can be granted, a certain amount of money must usually be available in the user's account, or other collateral must be proven. This is not the case with uncollateralized or undercollateralized loans. Here, the repayment of the loan in the same transaction is a condition for the actual disbursement. This means that the user only receives a loan if he pays it back almost immediately. Since no securities are required, the repayment can be limited on using the actual credit. In this way, the user has hardly any investment risks. Apart from the relatively small protocol fee, no own funds have to be used. This guarantees that the user will only receive a loan if he can achieve a positive return on his investment (or at least a zero-sum investment). Otherwise the loan cannot be repaid and the actual payout is not possible. In this way, no investment risk arises for the lender either. He never has to provide funds which are not paid back. On decentralised platforms, this enables almost risk-free Peer-to-Peer (P2P) lending without having to check collateral (Aave developers, 2020*b*).

## 1.2 Relevance of the topic

Flash loans are moving more and more into the public eye. As they are in contrast to traditional loans in their way of functioning, they offer a unique investment opportunity. But above all, the high profits that have been achieved through several attacks on the lending platforms by manipulating prices, provide the basis for exciting news. Nevertheless, there are almost no publications or researches worth mentioning on the subject of Flash Loans. This may be mainly due to the very young concept. News on the topic is mostly limited to the analysis of the attacks. The concept is usually only very superficially touched upon and there is no detailed analysis of further use of Flash Loans. Interesting aspects like

the functionality of the involved smart contracts of the lending platforms on the one hand, and detailed data analysis of the purposes of Flash Loans on the other hand are often not in focus. Reliable sources are thus limited to the description of the loan platforms (Aave developers (2020*b*), money-legos (2020), Uniswap (2020*b*)).

### 1.3 Objective

The central question of the present study can be derived from the observation of the increased interest in the topic of Flash Loans and the simultaneous lack of scientific reporting. It is about the question of which functionalities underlie the concept of Flash Loans. The following questions are the focus of the analysis: It will be elaborated how Flash Loans basically work and why they are a unique form of credit. This is connected with the description of the used smart contracts of the lending protocol of selected platforms. Furthermore, it will be described how the often reported attacks and the exploitation of large capital gains could come about. Furthermore, the present work focuses on different use cases and shows in an empirical analysis the extent of the current use of Flash Loans.

### 1.4 Methodology

Since there is hardly any scientific research on Flash Loans, the collection of the relevant data was done from scratch. For the acquisition of data from Aave the corresponding subgraph of TheGraph (2020) was used. Data about dYdX (2020) could be collected by querying the corresponding eventlogs via Etherscan (B1) (2020*c*). The identification of the relevant txns for flash swaps on Uniswap (2020*a*) was done using Dune Analytics (2020). Further txns for flash minting could be acquired through the respective contracts on Etherscan. A detailed description of the corresponding Application Programming Interface (API) queries can be found in the section Empirical research and the appendix. After iden-

tifying the relevant txns, the respective token transfers were drawn from Etherscan. By mapping known and labelled addresses from Etherscan token transfer, verified Etherscan contract addresses (Etherscan (B1), 2020a) and Qin et al. (2020) with the addresses of the token transfers (from, to, target, address, contractAddress) an overview of the origins and destinations of the transfers was created. This serves as a basis for being able to identify applications and involved protocols. Afterwards most common contracts used in txns were analysed on their application. Through this, many applications of Flash Loans could be identified directly. Furthermore txns which were not successful were checked for their error code to distinguish between the reasons of failure. Data acquisition, data cleansing and the identification of applications took the major part of the work.

## 1.5 Structure of the thesis

The introduction gives a first insight into the topic and presents the objectives of the work. Subsequently, in chapter two Flash Loans are distinguished from classic peer-to-peer lending and conceivable applications of Flash Loans are given. In addition the two comparable approaches flash minting and flash swaps are explained. Chapter three deepens the interaction of smart contracts with the protocols of the platforms Aave, dYdX, bZx, Uniswap and an example of flash minting. First the functionalities of the protocols which are key for the usage of Flash Loans are described in order to provide a good understanding of the mechanics. Following this, an explanation of the two famous flashloan attacks is given. Afterwards, chapter four offers an empirical analysis of the use of Flash Loans on the platforms presented. Finally, the discussion and evaluation of the results builds up the conclusion of the paper. Note that example transaction hashes, contract hashes, links to contracts, when the source is of an indicative nature only and is not a scientific source (like a post on twitter), as well as descriptive notes or statements which have no explicit quotable source (like personal notes by members of protocols) have been put into footnotes. All direct links have been marked in blue

for better visibility.

## 2 Decentralized loans

The concept flashloan (or Flash lending) was first introduced by Max Wolff (2018). However, the project around the open-source bank Marble has not yet seen a user-ready release. But even protocols for uncollateralized lending which are available live on the Ethereum main-net are currently not usable by the broad mass of DeFi-users. Decentralized Applications (dApps) such as DeFi Saver (2020) Collateral Swap (Kevin Truong, 2020) or flashmint (2020) try to fill this gap and offer a simple interaction with flashloan protocols or similar concepts. Nevertheless Flash loans like on Aave, dYdX or flash swaps on Uniswap are aimed at developers and technically experienced users. But not only the currently difficult interaction with flashloan protocols, but also the unique concept of loans without collateral on the borrower’s side poses challenges for users in understanding them. For this reason, the following chapter first distinguishes “classic” P2P lending, as it is mostly used for DeFi, from Flash Loans. Furthermore, an intuitive description for the functionality of Flash Loans is given. Chapter three goes into more detail about the used protocols and smart contracts.

### 2.1 Classic P2P lending

In order to explain the functioning of overcollateralized loans, the lending protocol of the Aave platform described in the whitepaper of Aave (2020a) offers a good starting point. The platform, which first started in 2017 under the name “Ethlend”, was launched in 2018. The lending protocol offers the possibility for decentralized pool lending and borrowing of cryptocurrencies. In this non-custodial protocol users can provide liquidity or borrow in over- or undercollateralized fashion. Depositors earn a passive interest on their loans. The protocol is based on a set of Smart Contract (SC). Deposits are paid into a pool contract. From this

pool contract, lenders can then borrow loans. The `LendingPoolCore` contract<sup>2</sup> stores the states of all reserves, i.e. the deposits of individual users. As already mentioned, there are two possibilities of borrowing. The first, overcollateralized loans, is the focus of this subsection. The second, Flash Loans, will be dealt with afterwards. Overcollateralized loans require users to lock a collateral with greater value than the loan in their reserves. Reserves represent the core of the pools. They hold the corresponding currencies which can be deposited and borrowed. Each reserve has a Loan-To-Value (LTV) which is a weighted average of the currencies held as collateral. The weighting is the corresponding value of the collateral in ETH. The amount a user can borrow depends on the currencies deposited still available in the reserves. A liquidation threshold value is set for every reserve. The threshold for a user is a weighted average of the thresholds of the currencies held as collateral. A loan can be liquidated when it reaches a healthfactor  $H_f < 1$  which is the dividend from the total amount of collateral weighted by the average liquidation threshold and the total amount of borrows including interest. If a collateral drops in price, causing the healthfactor to drop below one, then liquidators can buy the collaterals at a discounted price. The proceeds from the liquidation are then used to repay the loan until a maximum of 50%. The amount of collaterals will be liquidated so that the health factor rises above 1 again. The role of liquidator can be taken over by any user and thus acquire a part of the collaterals relatively cheaply. Users can take out loans at a fixed or variable interest rate and repay them in full or in partial amounts at any time (Aave, 2020a). The described functionality is similar in its basic features for many other crypto lending platforms. Cryptocurrencies are deposited as collateral and a loan for a certain amount of collateral can be obtained. In case of a decrease in value of the collaterals, a liquidation can be carried out in which other users can buy the collaterals at a reduced price to compensate a part of the credit of the user.

---

<sup>2</sup>[https://github.com/aave/\[...\]/lendingpool/LendingPoolCore.sol](https://github.com/aave/[...]/lendingpool/LendingPoolCore.sol)

## 2.2 Flash loans

The Ethereum Virtual Machine (EVM) contains the possibility to revert transactions if a certain condition is not fulfilled (Gudgeon et al., 2020). For this purpose a so-called checkpoint is set. Discarding a checkpoint throws away all changes that happened since that checkpoint. The EVM is reverted to this checkpoint if the following conditions of the smart contract are not fulfilled (Ethereum Foundation, 2020). This makes flashloan protocols possible. Since Flash Loans are uncollateralized loans, a user does not need a credit balance in his cryptowallet or even in the interacting SC, except for a small amount of the flashloan currency to pay the protocol fee (which also could be acquired through arbitrage gains). Note that network fees for initiating the txn are not taken into consideration here since they are accounted for every txn on Ethereum. When a user requests a flashloan, it is first made available to the SC. The SC of the user can then use the loan for various actions within the duration of the transaction and execute them. For example, it can be invested on other platforms. Different arbitrage concepts are conceivable here. Another possible application is debt refinance. For example, if a user has debt on one platform at a certain interest rate and wants to switch to another platform with a lower interest rate, one can use Flash Loans to clear the first debt and switch to another platform. The Applications section discusses the different variants of arbitrage and debt refinance.

After the execution of the SC actions, the user repays the flashloan plus a network fee. This is only possible if the user has made a profit through his SC. As there must not previously be a credit balance in the user's wallet, no other funds other than the actual loan can be used for repayment. However, if a User makes a loss on his investments, he can not pay back the loan. In this case the revert function mentioned above will be used. The transaction is put back into the state in which the loan has not yet been paid out. Since this all happens within one transaction, the actions of the SC are also reversed. So there is no investment on other platforms. This eliminates the risk of making a loss for both the user and the issuer of the flashloan.

## 2.2.1 Applications

Flash loans offer a wide range of possible applications whenever a user needs to have a large amount of cryptocurrencies at his disposal at short notice without having or being able to invest his own funds. The following section introduces them and gives examples for txns and projects for a specific application.

**Arbitrage** Flash loans theoretically offer every user the possibility to invest large sums of cryptocurrencies. For example, a loan from Aave is invested in another currency on a Decentralized Exchange (DEX). If the price of the currency is lower than the price on another platform, the currency can be sold there and an arbitrage profit can be made. An example of arbitrage using Flash Loans can be observed from a transaction on 18.01.2020<sup>3</sup>. The group “Flash Boys”, who boast of having carried out the first ever Flash Loan on the blockchain, were able to make a small profit on the transaction (Julien Bouteloup for Flash Boys, 2020). Figure 1 illustrates the transfers within the txn:

```
› From Aave: Lending Pool ... To 0x594f7f38c97a184... For 3,137.407014296228788899 ($3,226.28) 🏷️ Dai Stableco... (DAI)
› From 0x594f7f38c97a184... To Maker: Migration For 3,137.407014296228788899 ($3,226.28) 🏷️ Dai Stableco... (DAI)
› From Maker: Migration To 0x0000000000000000... For 3,137.407014296228788899 ($3,226.28) 🏷️ Dai Stableco... (DAI)
› From Maker: MCD Join SAI To 0x594f7f38c97a184... For 3,137.407014296228788899 ($3,186.31) 💎 Sai Stableco... (SAI)
› From 0x594f7f38c97a184... To Uniswap: SAI For 3,137.407014296228788899 ($3,186.31) 💎 Sai Stableco... (SAI)
› From Uniswap: DAI To 0x594f7f38c97a184... For 3,157.412913230790346677 ($3,246.85) 🏷️ Dai Stableco... (DAI)
› From 0x594f7f38c97a184... To Aave: Lending Pool ... For 3,148.38793884626558966 ($3,237.57) 🏷️ Dai Stableco... (DAI)
› From Aave: Lending Pool ... To 0xfac71f2395fd2a4... For 3.294277365011040228 ($3.39) 🏷️ Dai Stableco... (DAI)
```

**Figure 1:** Token transfers of first flashloan

Source: [txn](#):

[0x4555a69b40fa465b60406c4d23e2eb98d8aee51def21faa28bb7d2b4a73ab1a9](#)  
on Etherscan

To carry out the arbitrage trade, first a flashloan of 3,137.4070 DAI was made on Aave. After taken out the flashloan, DAI was converted into

---

<sup>3</sup>txn hash:

0x4555a69b40fa465b60406c4d23e2eb98d8aee51def21faa28bb7d2b4a73ab1a9



the same amount of Single-Collateral DAI (SAI) through MakerDAO's Migration contract by first burning the amount of DAI and then minting the same amount of SAI. Then SAI was converted into DAI using the platform Uniswap. It is important to note here that the amount of DAI is now higher than before the conversion. One can conclude here that there was a better conversion price of SAI to DAI on Uniswap than the one previously provided by Maker. The difference between the amount of DAI through the flashloan and the amount of 3,148.3879 DAI repaid to Aave was 10.9809 DAI. After repayment of the loan to Aave and deduction of the protocol fee of  $\sim 7.6867$  DAI, a profit of 3.2942 DAI was achieved (Qin et al., 2020). Despite the relatively small profit, this example of the first flashloan demonstrates the possibilities and functioning of arbitrage transactions using Flash Loans.

**Closing Collateralized Debt Position (CDP)** The standard in DeFi cryptocurrency lending is overcollateralized loans. This means that for a loan of a cryptocurrency a security in form of another currency must be deposited. The deposited collateral is then locked until the loan can be repaid (Aave, 2020a). In the case of very volatile cryptocurrencies, it is possible that the currency of the loan may fall sharply in value against the currency of the collateral. In this case, the borrower is unable to repay the loan, because the value of the currency is lower than the value when the loan has been taken, and his collateral remains locked. Flash loans offer the possibility to pay back the loan and thus release the collateral. For example, a borrower who has deposited ETH as collateral for a loan in the form of DAI can take out a flashloan in DAI in the amount of the CDP. With the flashloan he pays back the CDP and can release his collateral in ETH. Afterwards, the amount of ETH required to pay back the flashloan is converted into DAI by means of a decentralized exchange and the flashloan is repaid. Finally, the remaining ETH that was deposited as collateral is taken back<sup>4</sup>. An example is found in Figure 2:

---

<sup>4</sup>txn hash:

0x79644cd1bf45b196eb3aad9b96c1892568adc5ee830e746ed2d48d8a4081b2f2

- From Aave: Lending Pool ... To 0x8aaa0d145376ca... For 262.178591772051044931 (\$266.72)  Dai Stableco... (DAI)
- From 0x8aaa0d145376ca... To 0x00000000000000... For 262.178591772051044931 (\$266.72)  Dai Stableco... (DAI)
- From Maker: MCD Join E... To 0x8aaa0d145376ca... For 2.09506970387662191 (\$744.25)  Wrapped Ethe... (WETH)
- From Kyber: Reserve To Kyber: Contract For 264.82686037580913636 (\$269.41)  Dai Stableco... (DAI)
- From Kyber: Contract To 0x8aaa0d145376ca... For 264.82686037580913636 (\$269.41)  Dai Stableco... (DAI)
- From 0x8aaa0d145376ca... To 0x1e30124fde14533... For 2.412307871163245489 (\$2.45)  Dai Stableco... (DAI)
- From 0x1e30124fde14533... To Uniswap: DAI For 2.412307871163245489 (\$2.45)  Dai Stableco... (DAI)
- From 0x8aaa0d145376ca... To Aave: Lending Pool ... For 262.414552504645890871 (\$266.96)  Dai Stableco... (DAI)
- From Aave: Lending Pool ... To 0xe3d9988f6764571... For 0.070788219778453782 (\$0.07)  Dai Stableco... (DAI)

**Figure 2:** Token transfers of CCDP example

Source: [txn](#):

[0x79644cd1bf45b196eb3aad9b96c1892568adc5ee830e746ed2d48d8a4081b2f2](#)  
on Etherscan

In the example txn, a flashloan from Aave, equal to the outstanding debt position of  $\sim 262.1786$  DAI on the platform Maker is taken out. Next the CDP is paid back. For Maker this works by burning the amount (sending it to the origin address<sup>5</sup>). Then the collateral of  $\sim 2.0951$  Wrapped Ether (WETH) is freed and withdrawn. For paying back the flashloan, the user first converts the corresponding amount from the withdraw from WETH to DAI using the Kyber reserve. In the txn one can observe the difference between the required amount for repaying the flashloan and the converted amount of 2.45 DAI is sent to Uniswap, which may be a deposit that is not important for the process of CCDP. Following this, the flashloan of  $\sim 262.4145$  DAI, including a fee (read more in: 3.1 Aave Lending Protocol) is paid back and a small surplus is sent to the user's address. It is important to note here that the remaining amount of the collateral might have been withdrawn in another txn and is not observable within the present one.

DeFi Saver (2020) provides a dashboard with the same functions as described above for the platform MakerDAO. So it is possible to pay back loans and release locked collaterals without the need for additional own values in the form of cryptocurrencies.

---










<sup>5</sup>address:  
0x00

A similar principle as for closing CDP is also applied to the refinance of debt. If a user wants to switch from a DeFi platform with relatively high interest rates on his debt to a platform with lower interest rates, the use of Flash Loans is conceivable. First, a flashloan is taken out in the amount of the loan. The loan is then redeemed and the position closed. At the same time, the collateral deposited is released. Now the collateral is transferred to another platform and another loan is taken out at a lower interest rate. The flashloan is then repaid including the transaction fee. This makes it possible to change the interest rate of a loan without having to repay the loan from one's own funds. The costs incurred are only the transaction fees. In case of different cryptocurrencies in which the loans are issued, a currency swap can be made and changed into the preferred currency by using swap platforms such as Uniswap (Marc Zeller for Aave Blog, 2020).

A similar approach is followed for self liquidation. When a collateral drops in price, the users collateral is in danger of being liquidated. By observing the value of the collateral and noticing a sharp drop, a user can interact and buy back own collaterals. The procedure then follows the one described beforehand (Marc Zeller for Aave Blog, 2020).

**Collateral Swap** Collateral for loans may be held in multiple currencies. Aave, for example, offers a variety of currencies for this purpose. It may be desirable to change the currency in which the collateral is held. The amount of the loan depends largely on the performance and value of the collateral. In principle, increases in the value of currencies offer the possibility of obtaining higher loans with these currencies. In order to change from one collateral to another, it may be necessary to first redeem the entire loan and then take it up again with a new collateral. In order to avoid having to raise further own funds to repay the loan, a flashloan can also be used here. With a flashloan, a new currency is deposited as collateral and the other currency is withdrawn. The flashloan and the protocol fee are then repaid. Kevin Truong (2020) already is working on this concept under the name "Collateralswap" for the platform

MakerDAO<sup>6</sup>. An example is provided in Figure 3 below:

▶ From Aave: Lending Pool ... To CollateralSwap: Alpha For 25.000100426074895452 (\$25.42)  Dai Stableco... (DAI)  
 ▶ From CollateralSwap: Alpha To 0x00000000000000... For 25.000100426074895451 (\$25.42)  Dai Stableco... (DAI)  
 ▶ From Maker: MCD Join E... To CollateralSwap: Alpha For 0.5 (\$177.83)  Wrapped Ethe... (WETH)  
 ▶ From Uniswap: BAT To CollateralSwap: Alpha For 380.633221480682420826 (\$95.59)  BAT (BAT)  
 ▶ From CollateralSwap: Alpha To Maker: MCD Join B... For 380.633221480682420826 (\$95.59)  BAT (BAT)  
 ▶ From 0x00000000000000... To CollateralSwap: Alpha For 25.131350953311788653 (\$25.55)  Dai Stableco... (DAI)  
 ▶ From CollateralSwap: Alpha To Aave: Lending Pool ... For 25.087600777566157586 (\$25.50)  Dai Stableco... (DAI)  
 ▶ From CollateralSwap: Alpha To CollateralSwap: Dep... For 0.043750175745631067 (\$0.04)  Dai Stableco... (DAI)  
 ▶ From Aave: Lending Pool ... To 0xe3d9988f6764571... For 0.02625010544737864 (\$0.03)  Dai Stableco... (DAI)

**Figure 3:** Token transfers of collateral swap example

Source: [txn](#):

[0xf0c72b0e64612ba607d7499acc3dbf7dfcedc2a63901d38ec742eb64f1456ae4](#)  
on Etherscan





First, a flashloan of  $\sim 25$  DAI is taken out to collateral swap on behalf of the user. The user then withdraws all collateral on MakerDAO in form of 0.5 WETH using the flashloan. Then converts WETH for  $\sim 380.64$  Basic Attention token (BAT) using Uniswap (read more in: 3.4.1 FlashSwaps). Afterwards the user borrows  $\sim 25.1314$  DAI from MakerDAO by using BAT as collateral. The flashloan of  $\sim 25.0876$  DAI is paid back and a surplus of  $\sim 0.02625$  DAI is sent to the user. This way the user swapped the collateral from WETH to BAT in only one txn without using own funds.

**Wash trading** The following use case briefly presents a deliberate way of manipulating trading volumes. Wash trading describes the procedure of exchanges to influence the market by inflationary trading of certain assets. Wash trading gives investors the illusion of a greater trading volume of certain assets than would actually be traded on the open market. It “therefore distorts demand and supply of cryptocurrencies and artificially affects the price and harms the development of the cryptocurrency market in the long run” (Cong et al., 2019). Alongside with this

<sup>6</sup>txn hash:

[0xf0c72b0e64612ba607d7499acc3dbf7dfcedc2a63901d38ec742eb64f1456ae4](#)

statement, Cong et al. (2019) find evidence that a large proportion of unregulated exchanges inflate trade volume through computer programs. The Surveillance Report of the Blockchain Transparency Institute (2019) estimates that the wash trading rates of the major cryptocurrencies range from 50% to 75%<sup>7</sup>. According to Qin et al. (2020) the biggest hurdle to wash trading is the large volume of proprietary cryptocurrencies needed to signal a sufficient percentage increase in trading volume. Especially for DeFi exchanges, which settle all transactions on-chain and can not use off-chain transactions for the pretence of higher volumes, this restriction is binding. With the help of Flash Loans it is possible to reach very high trading volumes in a very short time without having to raise substantial own funds<sup>8</sup>. Figure 4 shows an example wash trading txn.

▶ From dYdX: Solo Margin To 0x8687d8765a3948... For 0.01 (\$3.57)  Wrapped Ethe... (WETH)  
 ▶ From Uniswap: LOOM To 0x8687d8765a3948... For 122.189781507534529628 (\$2.67)  Loom (LOOM)  
 ▶ From 0x8687d8765a3948... To Uniswap: LOOM For 122.189781507534529628 (\$2.67)  Loom (LOOM)  
 ▶ From 0x8687d8765a3948... To 0x3944a4ec95fba5e... For 0.009940475558103003 (\$3.55)  Wrapped Ethe... (WETH)  
 ▶ From 0x3944a4ec95fba5e... To dYdX: Solo Margin For 0.010000000000000001 (\$3.57)  Wrapped Ethe... (WETH)

**Figure 4:** Token transfers of wash trading example

Source: [txn](#):

[0xf65b384ebe2b7bf1e7bd06adf0daac0413defeed42fd2cc72a75385a200e1544](#)  
on [Etherscan](#)

In this txn a flashloan on dYdX is carried out (read more about dYdX in: 3.2 dYdX). Then 0.01 WETH was converted into  $\sim 122.18978$  LOOM and back into  $\sim 0.0099$  WETH. Afterwards the flashloan was paid back (Qin et al., 2020). What distinguishes this txn from other applications, such as arbitrage, is that the txn has been carried out although there was a loss when swapping tokens. This indicates that the user wanted the txn to complete to push up trading volumes and did not try to make a profit from it in the first place. Qin et al. (2020) state that “the 24-hour trading volume of the ETH/LOOM market increased by 25.8% (from 17.71 USD to 22.28 USD) as a result of the two trades”.

<sup>7</sup>Bitcoin: 50%, Ethereum: 75%, Ripple (XRP): 55%, and Litecoin: 74%

<sup>8</sup>txn hash:

[0xf65b384ebe2b7bf1e7bd06adf0daac0413defeed42fd2cc72a75385a200e1544](#)

## 3 Flash loans in detail

The following chapter goes into detail about the smart contracts of the platform Aave, dYdX, comparable approaches like Uniswap and flash minting and the interaction with them. Based on this, the attacks are described which have brought Flash Loans into the focus of the public and media. It is explained how the attackers were able to achieve such high profits by exploiting SC vulnerabilities and manipulating Oracle contracts.

### 3.1 Aave Lending Protocol

The following section introduces the interaction of SC with the Aave protocol and gives examples.

Aave developers (2020a) offers a good introduction to setting up a SC to perform a flashloan. The Simple flashloan SC framework in the appendix provides the basic framework for executing a flashloan.

`ILendingPoolAddressesProvider`, `FlashLoanReceiverBase` and `ILendingPool` are provided by mrdavey for Aave (2020).

In the contract definition, a constructor argument is passed to the inherited `FlashLoanReceiverBase` contract which determines which `LendingPoolAddressProvider` is used on the Network. The address has to be changed according to the network on which the contract is deployed. Here DAI on Ethereum Mainnet is used. The shown contract calls the `LendingPool` contract of Aave, requesting a flashloan through the call of the function `flashloan()`. It is set up to only let the owner of the contract execute a flashloan of the stated amount of DAI. It is important to notice that 1 ether refers to 1'000 tri WEI worth of 1 DAI. The argument `_params` lets one define additional parameters. The requested amount is then transferred to the contract by the `LendingPool` out of the `reserve`. It then calls the `executeOperation()` of the receiving contract specified in `contractdefinition`. The `_reserve` corresponds to the address of the underlying asset as defined in the `address` in the

`flashloan` function. `_amount` is the borrowed amount of the loan with a `_fee` of 0.09% to be paid back. Next any application specified within the contract is carried out and the flashloan is transferred back to the `LendingPool` contract. This contract compares the balance of the `reserve` before and after the execution of applications<sup>9</sup> to be the same amount plus the flashloan fee. The `totalDebt` is defined as the amount of the loan plus the `_fee`. After compiling and deploying the contract to Ethereum mainnet, an amount of DAI sufficient of the fees has to be transferred to the contract. If this is not done, the txn is reverted and will fail because the available liquidity after execution of applications is less than the liquidity before plus the flashloan fee.

## 3.2 dYdX

The logic of Flash Loans on dYdX follows in principle the same as on Aave. However, dYdX has no native flashloan feature integrated. The solo protocol consists of SC on Ethereum which support margin trading, borrowing, and lending (dYdX, 2020). It is account based and all state changes of one or multiple accounts happen through so called “Actions”. Actions support all common transactions such as deposit, borrow, trade, transfer, buy, sell, call, liquidate and vaporize<sup>10</sup>.

To perform more complex operations, it is possible to combine several actions into operations. The most important feature here is that collateralization of the accounts is only checked at the end of each operation. This means that multiple actions can be performed without the need for collateralization per se. For example, loans can be taken out, used for different applications, and only need to be repaid at the end of an operation. The logic of Flash Loans on dYdX follows the same as on Aave (money-legos, 2020). The difference lies in a much lower fee of only 2

---

<sup>9</sup>`availableLiquidityAfter == availableLiquidityBefore.add(amountFee),`  
(Aave, 2020b)

<sup>10</sup>Pulls funds from the insurance fund to recollateralize an underwater account with only negative balances (dYdX, 2020)

WEI. Despite this statement, fees of 1 WEI could be observed<sup>11</sup>, and a statement of dYdX founder Antonio Juliano in a personal discussion<sup>12</sup> suggests that there are no fees at all.

### 3.3 bZx

The protocol bZx also contains an explicit flashloan feature through the function `flashBorrowToken` within the `SC LoanTokenLogicV4`<sup>13</sup> (CertiK, 2020a). Although there is only one observable flashloan `txn`<sup>14</sup> that call the function and then perform actions the functionality will be pointed out here for the sake of completeness. The only executed flashloan on bZx is the second famous flashloan attack, which will be presented later. BZx “is a decentralized protocol that enables lending and borrowing for margin trading” (bZeroX, LLC, 2018). For this purpose, bZx uses global liquidity pools which are shared between many different exchanges (Hussey and Tran, 2020). When users deposit liquidity they receive `iTokens`, which serve as interest bearing asset. These tokenized loans reward lenders with interest proportional to the amount of `iToken` held by each lender. They are minted by the contract mentioned before by calling the function `mint` equivalent to an amount of asset deposited. When borrowing assets (to trade with margin), users borrow from an `iToken` lending pool and receive the corresponding underlying asset. When performing a flashloan, the function `flashBorrowToken` first performs a reentrancy check to prevent potential attackers from entering the initial contract again and calling the function before it has been completed (fravoll, 2019). A pausing check validates that the function has not been paused by an admin. The function then transfers the `borrowAmount` of the asset to the address of the `borrower` and then might send it to the `target` address by an `ArbitraryCaller.sendCall` within the contract `ArbitraryCaller`. The target can be an external address or contract

---

<sup>11</sup>`txn: 0xc9c1d6563f2db590c8786ad98ce07daddf9bd1e753b57ed695fe9475d234ffa5`

<sup>12</sup>On Discord 11.07.2020

<sup>13</sup><https://github.com/bZxNetwork/.../LoanToken/LoanTokenLogicV4.sol>

<sup>14</sup>`txn:`

`0x762881b07feb63c436dee38edd4ff1f7a74c33091e534af56c9f7d49b5ecac15`



which then performs some applications with the flashloan. Afterwards, the remaining asset balance is transferred back and a check on the asset balance is made.

```
require(  
    address(this).balance >= beforeEtherBalance &&  
    ERC-20(loanTokenAddress).balanceOf(address(this))  
        .add(totalAssetBorrow) >= beforeAssetsBalance,  
    "40"  
);
```

Code 1: Requirement of function `flashBorrowToken` within `LoanTokenLogicV4`

The check requires the ETH balance before the loan not to be less than after and the ERC-20 token balance of the asset of the loan added up by the amount borrowed not to be less than the balance before.

## 3.4 Comparable approaches

The following section describes the concepts flash swaps and flash minting which follow the idea of Flash Loans. Although they are not explicitly assigned to the term Flash Loans because flash swaps technically only involve the exchange of two tokens and flash minting does not request a loan, the possible applications are the same. The general functioning of the Uniswap protocol is explained because it is key to understanding arbitrage concepts and flashloan attacks presented later.

### 3.4.1 FlashSwaps

With the launch of version 2 of Uniswap (2020*b*) the feature flash swaps was introduced. In contrast to a regular swap, where one token is exchanged for another, flash swaps allow the withdrawal of the token without a prior check on the corresponding amount of the other token to be exchanged. Uniswap allows users to withdraw the entire available amount of a token's liquidity pool without having to show the corresponding amount to be exchanged. The logic again follows the logic of Flash Loans. A user can dispose of the amount of the received tokens

within a transaction as long as he repays or exchanges them at the end of the transaction including a fee of  $\sim 3\%$ . Uniswap V2 is a DEX based on two SC on the Ethereum network (Uniswap, 2020g). It allows the swap of ERC-20 token pairs and the swap of ERC-20 and ETH. The two main SCs involved are a Factory contract and a Pair contract. The UniswapV2Factory SC contains an exchange registry where a token address is linked to the address of an UniSwapV2Pair contract. By calling the `createPair` a Pair contract for a particular ERC-20 token is deployed. In order not to create Pair contracts for tokens that already exist, a comparison is made in the registry. A user can then invoke the `addLiquidity` function within a router contract to add liquidity to the respective address linked to the token. Routers are helper contracts “that perform the safety checks needed for safely swapping, adding, and removing liquidity” (Uniswap (2020f), Uniswap (2020e)). Each Pair contract contains a liquidity pool for exactly one ERC-20 token and ETH or two specific ERC-20 tokens (in the following the term “token” will be used representative for any ERC-20 token or ETH). The exchange price for the included tokens is determined by the ratio  $x \cdot y = k$  between token amount  $x$  and token amount  $y$  (Uniswap, 2020c). The ratio is kept constant during swaps and changes only when liquidity is added to a pool by calling the function `addLiquidity`. During a swap a user receives the amount of tokens needed to keep the relation constant after a certain amount of tokens has been deposited. Because the ratio of tokens changes, the exchange rate changes as well and gives incentives to swap in the opposite direction. When liquidity is added to a pool for the first time, the user has to decide which rate between tokens is considered appropriate. Subsequent liquidity providers must adhere to the specified ratio at the time of deposit. If the ratio is different from the general market price, arbitrage possibilities arise through swaps, driving the exchange rate back to a level near the market price. To incentivize users to provide liquidity the function `addLiquidity` within the Pair contract mints an amount of ERC-20 liquidity tokens respective to the amount of another token added. Liquidity tokens can be traded or burned to remove liquidity. When burned, a user receives the respective amount of ether.

ETH and ERC-20 tokens are withdrawn at the current exchange rate, not the initial rate at the deposit, which facilitate arbitrage possibilities when burning liquidity tokens. V2 of the Uniswap protocol introduces the possibility of flash swaps (Uniswap, 2020*d*). To be technically correct, all swaps occurring on V2 are flash swaps. This is because it is never required for a user to deposit the amount of one token or ETH needed for a swap before receiving the corresponding amount of token, as long as it is deposited within the same txn (which does not necessarily mean that users more often do not deposit before they take out swaps). To initiate a swap the function `swap` is called.

```
function swap(uint amount0Out, uint amount1Out, address to, bytes
    calldata data);
```

Code 2: function `swap` (Uniswap, 2020*d*)

`amount0Out` and `amount1Out` determine the amount of the two tokens included in the token pair that `msg.sender` wants to send to the `to` address. In a classical swap a user usually sends the appropriate amount to the Pair contract before calling the `swap` function. To determine between a usual swap and a flash swap the parameter `data` is used. For `data.length=0` the contract assumes that the payment of one token has already been made and the tokens are transferred directly to the `to` address. In case of `data.length>0`, the contract calls the function `uniswapV2Call` which has to be included in an external SC on the `to` address but also transfers the tokens to this address.

```
1 function uniswapV2Call(address sender, uint amount0, uint amount1, bytes
2     calldata data) {
3     address token0 = IUniswapV2Pair(msg.sender).token0(); // fetch the
4     address token1 = IUniswapV2Pair(msg.sender).token1(); // fetch the
5     assert(msg.sender == IUniswapV2Factory(factoryV2).getPair(token0,
6         token1)); // ensure that msg.sender is a V2 pair
```

Code 3: function `uniswapV2Call` (Uniswap, 2020*d*)

`sender` corresponds to `msg.sender`, `amount0` and `amount1` to the respective `amount0Out` and `amount1Out` of the token pair of the swap. Line 2 and 3 fetch the addresses of the tokens included in the pair while line 4 checks if `msg.sender` is an actual Uniswap V2 pair address. After the

call of `uniswapV2Call` the respective amount of tokens including a fee of must be transferred back. The fee of withdrawing tokens on Uniswap is 3%, whereas the amount in the reserve after the return must be greater than before:  $TokenReservePre - TokenWithdrawn + (TokenReturned \cdot .997) \geq TokenReservePre$ . This is because Uniswap calculates fees on the input amount.

### 3.4.2 Flash-minting

Probably the most complex concept requires a short introduction to token minting. Mintable tokens are ERC-20-compatible tokens. These can be created at any time in unlimited numbers, for example by the tool ERC20 token generator (2020) and added to the total supply. They differ from standard ERC-20 tokens which have a fixed total quantity (tokenmint, 2019). Self-made tokens have no monetary equivalent at first. But certain tokens like the stablecoin DAI must be deposited with collaterals before minting, which correspond to the equivalent value of the new tokens in USD. In this way, DAI can maintain its coverage with the USD. The concept flash-minting uses the described creation of new tokens. Qin et al. (2020) describe flash-minting as the idea to allow an instantaneous minting of an arbitrary amount of an asset. The only condition is that all new tokens are destroyed within the transaction in which they were created. Economically, this only makes sense when minting asset backed tokens. These are 1:1 backed by another non-mintable currency. For example Waped Ether (WETH) is 1:1 backed by Ether. Thus mintable tokens can represent an economic value. One WETH can also be transferred and traded at any time and always retains the equivalent value of one ETH. Flash-mintable asset-backed tokens can be changed into the underlying asset and anyone can mint arbitrarily many unbacked tokens and spend them at full face value, so long as they destroy all the unbacked tokens (and therefore restore the backing) before the end of the transaction (Austin Williams, 2020). Despite the possibility to mint an unlimited number of tokens, they can still be accepted as the underlying asset. The logic is the same as for Flash Loans. All applications men-

tioned above are conceivable here. The difference with flash minting is that the source of funds is not a lending platform like Aave. However, the procedure is analogous. A minimal example contract by Austin Williams (2020) for his project “FlashWETH” is given in the following:

```
pragma solidity 0.5.16;

import "@openzeppelin/contracts/token/ERC-20/ERC-20.sol";

interface IBorrower {
    function executeOnFlashMint(uint256 amount) external;
}

contract FlashMintableToken is ERC-20 {
    function flashMint(uint256 amount) public {
        // mint tokens
        _mint(msg.sender, amount);
        // hand control to borrower
        IBorrower(msg.sender).executeOnFlashMint(amount);
        // burn tokens
        _burn(msg.sender, amount);
    }
}
```

Code 4: Minimal Flash Minting Example (Adapted from Austin Williams (2020))

The function `flashMint` is called to mint the tokens. Afterwards another function has to be executed to use the tokens for applications. In the end the same amount of tokens is burned.

### 3.5 Flash loan attacks

In the following, the two largest flashloan attacks are described and it is explained how the attackers could make  $\sim 350k$  USD and  $\sim 650k$  USD profit. The mechanics of the attacks have been shown quite well in various analyses (PeckShield (2020), Auguste (2020)) and a reproduction would exceed the scope of the present work and would not provide added value for the economic analysis of Flash Loans. Therefore, the following section aims to provide an understanding of why Flash Loans were used as a basis for the attacks.

The first attack<sup>15</sup> took place on 2020-02-15 01:38:57 and differs from the

---

<sup>15</sup>txn:

0xb5c8bd9430b6cc87a0e2fe110ece6bf527fa4f170a4bc8cd032f768fc5219838

attack described later because no oracle manipulation was performed, but a bug in a SC of a DEX (bZx) was taken advantage of (PeckShield, 2020). After taking up a flashloan on dYdX for 10,000 ETH, 5500 ETH were first deposited on Compound as collateral to borrow 112 WBTC. Compound offers the possibility of earning interest by depositing into liquidity pools and borrowing assets (Leshner and Hayes, 2019). When users deposit assets they receive cTokens representing the Asset. In the present txn this is observable by 274,843.67745507 cETH. When users borrow, they lock their deposits as collateral and receive a specific share of the collateral in form of the borrowed asset, determined by a collateral factor of the pool from which is borrowed. Next, the attacker opened up a short 5x position on WBTC by sending 1300 ETH to bZX. In the presented attack, the user received “sETHwBTC5x”-tokens, which indicates a deposit of ETH for 5x WBTC and called for a swap of ETH to WBTC. BZx leverages swaps through Kyber Network. Kyber Network (2019) provides liquidity from different resources. When users request trades for a token pair, Kyber searches reserves to find the best price for the trade or the reserve which is capable of providing the liquidity required for the trade. The only pool which had enough liquidity to provide such a large amount was Uniswap, so the margin trading function leveraged KyberSwap to swap 5637.623762 ETH for 51.345576 WBTC (PeckShield, 2020). At this point, it is considered that Uniswap could not provide enough liquidity to service the entire amount. Because the ratio of tokens within the Uniswap liquidity pool of the token pair ETH/WBTC changed (see 3.4.1 FlashSwaps), the exchange rate changed from  $\sim 38.5$  WETH/WBTC to  $\sim 109.8$  WETH/WBTC. This part of the transaction was only executed to manipulate the conversion rate on Uniswap. WBTC was purchased at an extremely cheap price of 0.00910766 WETH/WBTC, while the normal market price was  $\sim 0.0275$ . This normally results in a holding of WBTC which is worth less than the swapped WETH under non-manipulated market prices. At this point, if the price fall is more than 20%, a requirement within the `iTokens_loanOpeningFunctions`<sup>16</sup> should actually intervene to revert the call and prevent the bZx liquidity

---

<sup>16</sup>[https://github.com/bZxNetwork/.../iTokens\\_loanOpeningFunctions.sol](https://github.com/bZxNetwork/.../iTokens_loanOpeningFunctions.sol)

pool from ending up with less liquidity than is covered by collaterals.

```
1  require ((
2      loanDataBytes.length == 0 && // Kyber only
3      sentAmounts[6] == sentAmounts[1]) || // newLoanAmount
4      !OracleInterface(oracle).shouldLiquidate(
5          loanOrder,
6          loanPosition
7      ),
8      "unhealthy position"
9  );
```

Code 5: require in `iTokens_loanOpeningFunctions.sol`

The attacker managed to bypass this check by making sure that `loanDataBytes` is empty. Since `sentAmounts[6] == sentAmounts[1]` is true because it guarantees that the amount of the loan equals the amount to withdraw, the oracle was never consulted and there was no check for liquidation (Livnev, 2020). This kind of strategy, raising the margin over the rate which could be realized when market prices would operate without being manipulated, is known as “margin pump”. Afterwards the pump, the attacker sold his 112 WBTC from Compound to Uniswap using a regular swap. The price realized in the swap was 61.4 WBTC/ETH, leading to a return of 6871.4127388702245 WETH. The flashloan was paid back, using the remaining 3200 ETH from the initial loan and  $\sim 6800$  ETH of the swap return (plus a small fee), leading to a profit of  $\sim 71.4174$  ETH for the attacker.

Auguste (2020) states that the second attack<sup>17</sup>, which happened on 2020-02-18 03:13:58, was using the same idea of manipulating prices but with a different method. At first a flashloan was taken up by the function `flashBorrowToken` of bZx. Note that the only flashloan ever made on bZx was made by the attack. The reasons can be found in the found in the pause of the protocol and because the functionality is poorly documented and not explicitly advertised by the platform. The procedure again uses Kyber to manipulate prices. For this purpose, the flashloan is used to make several purchases of sUSD (synth USD). sUSD is a coin minted against the value of USD by the platform Synthetix (2020). Synthetix functions as a liquidity provider very similar to Uniswap and forms

---

<sup>17</sup>txn:

0x762881b07feb63c436dee38edd4ff1f7a74c33091e534af56c9f7d49b5ecac15

one of the reserves for Kyber. Kyber selects the reserve of the platform with the best available price or the ability to service the requested amount. For this purpose, 540 ETH were first converted on Uniswap Kyber reserve and then 20 ETH on sUSD Kyber reserve were converted to sUSD. This was done 18 times and each time the conversion price got worse from  $\sim 262.0492$  ETH/USD to  $\sim 111.2258$  ETH/USD as the liquidity in Kyber reserves decrease. bZx uses Kyber as price oracle. Due to the manipulated prices of the Kyber Reserves, the Oracle reports a price which differs from the actual market price. The attacker next exchanges  $\sim 3,517.8591$  ETH for  $\sim 943,837.5876$  sUSD on Synthetix. The amount attended to exchange have been 6000 ETH, but Synthetix did not seem to have enough liquidity at that time. The remaining ETH was transferred back. The price of  $\sim 268,2989$  ETH/USD is significantly above the manipulated price reported by the Kyber oracle. Next, sUSD on bZx is exchanged for ETH. The sum of  $\sim 1,099,841.3921$  sUSD is composed of all swaps on Kyber reserves and the exchange on Synthetix. The attacker can thus obtain  $\sim 6796,0128$  ETH at a price of  $\sim 161,8362$  ETH/sUSD. This is nearly 100 USD less than the price called at the beginning of the attack. Since sUSD follows the value of the USD, bZx sUSD worth  $\sim 1,099,841,3921$  USD, while it returns ETH worth  $1,780,889,52138$  (calculated at the price at the beginning of the attack) USD. This represents a loss of  $\sim 681,048,1293$  USD. After the exchange, the attacker repays the flashloan consisting of  $\sim 6796,0128$  ETH plus remaining  $\sim 3082,1409$  ETH from the initial flashloan, leading to a profit of  $\sim 2378,1537$  ETH.

While the first attack exploited a bug in a function of the bZx protocol that did not contact the oracle, the second attack was aimed at getting the oracle itself to deliver manipulated prices. Even if manipulative and legally questionable, both attacks show the possibilities of Flash Loans. They both took advantage of the opportunity to take out very high loans in a very short time without putting themselves at investment risk. These examples show that SC need to be better audited to be prepared for the fact that with Flash Loans, there now is theoretically a possibility for all DeFi users to execute txn with high sums. Further more precautions in



preventing oracle manipulation must be taken.

## 4 Empirical research

The following chapter analyses the use of Flash Loans. The analysis focuses on the frequency of use of Flash Loans, the used decentralized protocols on which they are executed and the amount of the loans. The main interest lies in the categorization of use cases. Criteria are defined how Flash Loans and the associated transactions can be categorized into use cases. The aim is to find out which of the above mentioned possible applications are actually used. It is of interest how high the share of Flash Loans on the presented protocols is and which volumes in USD are observable.

The chapter first describes the methodology of data collection and preparation, then goes into the categorization of use cases and concludes with the presentation of the results.

### 4.1 Data acquisition and preparation

TheGraph (2020) protocol is used for data collection on Aave. It allows to run GraphQL (The GraphQL Foundation, 2020) queries on Web3 dApps. This can be done by a simple API-call. The advantage is that it is possible to get a large amount of information from the Ethereum block chain with a single call.

The data collection for Flash Loans on dYdX turned out to be more complex, since no subgraph exists for dYdX Flash Loans and the definition of an own subgraph, due to missing official definition of Flash Loans on dYdX, turned out to be less reasonable. By examples of Flash Loans on dYdX<sup>18</sup> the following regularities could be determined: Within a transac-

---

<sup>18</sup>txn hashes:

0xf65b384ebe2b7bf1e7bd06adf0daac0413defeed42fd2cc72a75385a200e1544  
0xb5c8bd9430b6cc87a0e2fe110ece6bf527fa4f170a4bc8cd032f768fc5219838

tion, applications of Flash Loans, i.e. various token transfers, are always included in the eventlogs `LogWithdraw`<sup>19</sup> and `LogDeposit`<sup>20</sup>. The corresponding eventlogs are exclusively assigned to the `SoloMargin`<sup>21</sup> SC of dYdX. To obtain the relevant txns to be considered Flash Loans, all txns of the contract containing the two eventlogs were identified. However, during the data analysis it was found that txns which include the simultaneous deposit and withdraw, but not in the form of a flashloan, also meet the conditions stated before. This is the case, for example, when a user deposits a sum of  $x$  US Dollar Coin (USDC) and borrows a sum of  $y$  DAI in the same txn<sup>22</sup>. As a result of this observation, txns containing the transfers “withdraw” and “deposit” in which the amount withdrawn and the amount deposited do not correspond were removed. This is also true for the two events when two different tokens are involved. The difference in amounts to be accepted was set to 2 WEI because different sources report different fees (see: 3.2 dYdX)

To identify FlashSwaps, Uniswap transactions were considered which contain a `length.data` argument  $> 0$  for the function `swap`. This is because the `data` parameter will return a `data.length = 0` only if the Pair contract has previously been paid with the appropriate amount of one token in the token pair before the swap is executed to equalize the pair. All returns  $> 0$  are flash swaps because the swap took place before a corresponding amount of another token was deposited. The following query on Dune Analytics (2020) returns the hashes of the matching txn.

23

```
SELECT call_tx_hash
FROM uniswap_v2."Pair_call_swap"
WHERE octet_length(DATA) > 0
LIMIT XYZ;
```

Code 6: Dune Analytics Query FlashSwaps

The call is pretty self-explanatory. `"call_tx_hash"` returns a list

<sup>19</sup>Log Hash: 0xbc83c08f0b269b1726990c8348ffdf1ae1696244a14868d766e542a2f18cd7d4

<sup>20</sup>Log Hash: 0x2bad8bc95088af2c247b30fa2b2e6a0886f88625e0945cd3051008e0e270198f

<sup>21</sup>contract: 0x1e0447b19bb6ecfdae1e4ae1694b0c3659614e4e

<sup>22</sup>e.g.: txn: 0x104a67d9e03081f81e4e67e25f6b4bdeb237a752c97797ed1403d09cd48731b7

<sup>23</sup>According to Noah Zinsmeister of Uniswap in a personal discussion on discord 15.07.2020

of the respective txn hashes, "uniswap\_v2.Pair\_call\_swap" refers to the Uniswap swap function when a `data.length > 0` is returned (`"octet_length(DATA) > 0"`). Additionally a limit can be set.

The identification of flash minting txn has been done by the two contracts `FlashWeth`<sup>24</sup> and `FlasMintableETH`<sup>25</sup> by Austin Williams (2020).

Data has been collected between block 7979145 (Jun-18-2019 00:29:13 AM) and block 10464540 (Jul-15-2020 02:11:21 PM). After data cleansing the time frame was reduced to block 9142247 (Dec-21-19 08:04:25 PM) until block 10464540 (Jul-15-2020 02:11:21 PM).

The initial amounts of the Flash Loans on Aave could be taken directly from TheGraph call. For dYdX the transaction action pairs "Borrow" and "Repay" have been compared and the Flash Loan amount taken was set to those ones with the same amount (plus a fee tolerance of up to 2 WEI) and the same token. This has been done with all possible care to avoid that actions that represent a borrow and repay within the Flash Loan txn are included in the analysis. If there were multiple candidates, the pair that brackets other pairs was considered. For Uniswap the amounts of the transaction action "Swap amount token on Uniswap" has been taken into consideration. Further the identified amounts have been controlled to be transferred from UniSwapV2 because the Flashswap feature is only available in V2.

For the identification of applications of all token transfers of the txn identified as Flash Loans, flash swaps or flash minting were loaded from Etherscan. A description of all used API-calls can be found in the appendix "API Calls". After the collection of all data, as many known addresses as possible were assigned to the token transfers that take place within the txn. For this purpose the addresses labelled by Etherscan's token transfers, verified Etherscan contract addresses (Etherscan (B1), 2020a) and Qin et al. (2020) were used. These were mapped with the addresses of the token transfers to get a good readable overview of possible

---

<sup>24</sup>contract: 0xf7705C1413CffCE6CfC0fcEfe3F3A12F38CB29dA

<sup>25</sup>contract: 0x09b4c8200f0cb51e6d44a1974a1bc07336b9f47f

applications.

## 4.2 Categorization

After the data acquisition, the same or similar transfers and txns were grouped. The following chapter presents the methods used for the identification of applications. For this purpose, it shows identified subcategories of applications, if applicable, and describes the path of identification. When a txn could be identified, the corresponding smart contracts txns and the contracts deployed by the owner address have been checked for similar txn schemes. Subsequently, a cross-check was carried out on a selection of txn of a contract to see whether they follow the scheme of the application. If this was the case, all txn which were included in the contract were assigned to the found application. The corresponding subcategories are assigned to the txns in the data set. It is important to notice that there is no deep explanation about the functioning of said subcategories because the complexity of involved txns would lead the explanations to exceed the scope of the present work by far. Instead, a short explanation is given. Examples can be found within the data set under the respective “subcat” tag. When more than one use cases within an application could be determined (e.g. see: Arbitrage\_hammerbots 4.2) an entry in column “subcat2” has been made.

### Arbitrage

**Arbitrage\_bots** From a known arbitrage bot<sup>26</sup> it could be identified that several contracts deployed by the owneraddress work the same way. The contracts basically take out a flashloan and try to make profit by a combination of buying and selling currencies on DEX, in combination with staking these currencies, for favorable prices.

---

<sup>26</sup><https://0xtracker.com/traders/0x1d92ed2c7cb9bb19c654dcec059d043856b601e4>

**Arbitrage\_swaps** The DEX “1inch” reported an arbitrage trade of Chi Gastoken on Twitter<sup>27</sup>. Said contract swapped Gastoken minted by 1inch on Uniswap for a higher price to WETH for a profit of  $\sim 0.07584$  WETH. Other txns by the same contract show similar behaviour by swapping tokens for higher prices.

**Arbitrage\_HXY** A variation of arbitrage is shown by using the platform Hex (2018 -2020) in combination with Flash Swaps. The platform allows earning and staking of the token HEX by depositing ETH. Additionally, there is an ERC-20 incentive token called HEXMONEY (HXY). HXY can be minted by a referral program or by a transfer and conversion from HEX to the hex.win platform. Although some sources assume a scam behind the platform (see: Nikolaev (2019)) it is possible to trade HEX as well as HXY or to swap on Uniswap<sup>28</sup>. The observed arbitrage strategy takes advantage of the minting of 1 HXY per 1000 HEX transferred to hex.win and is structured as follows:<sup>29</sup>:

A flash swap over a high sum of 31337 HEX is taken. HEX is transferred to Hex.win and converted to HXY. The origin address<sup>30</sup> mints the transferred amount of 1 HXY per 100 HEX and an additional amount of 1 HXY per 1000 HEX making up  $31.337 + 3.1337 = 34.4707$  HXY to the user. The user pays the flash swap by the tokenpair HXY/HEX and keeps the amount of additionally minted HXY. Again, multiple txns of the same contract could be identified because they execute similar transfers and thus belong to the same category.

**Arbitrage\_comp** A similar approach to the one before involves provision of liquidity loan to the protocol Compound Labs, Inc. (2020) and receiving the incentive (or government) token “COMP”. This procedure is

---

<sup>27</sup><https://twitter.com/1inchexchange/status/1265198816619266048>

<sup>28</sup><https://uniswap.info/token/0x2b591e99afe9f32eaa6214f7b7629768c40eeb39>

<sup>29</sup>txn:

0x0a070096bd56a8aaab31370f1f3afeaad0e8ace4b755f206983ddcbc24d772e1

<sup>30</sup>address:

0x00

called COMPfarming and can be observed in various flashloan txns via a contract. Users can receive COMP by supplying, borrowing, withdraw, or repaying an asset (Compound Labs, Inc., 2020). This incentivizes users to interact with Compound frequently. Users take out a flashloan and carry out as many of the above interactions with Compound as possible. The resulting COMP tokens can then be sold and an arbitrage profit is generated. All txns interacting with the contract `COMPfarming` has been assigned to this subcategory.

**Arbitrage\_liquidation** The presented arbitrage option liquidates accounts whose collaterals have declined in value. First, a flashloan of the amount of the collateral to be liquidated is taken out and then the same amount is liquidated. The liquidator buys the outstanding amount at a lower price, which results in a higher return of the same token. The difference between the flashloan and the received amount of the token is an arbitrage profit. Txns following the same purpose could be identified by the parameters “Liquidator Repay” and “Liquidation” as found in the transaction actions on Etherscan<sup>31</sup>.

**Arbitrage\_dao** “Arbitrage DAO is a DeFi Union Arbitrage Fund, [...] using a combination of on-chain liquidity and off-chain bots designed for arbitrage opportunities” (Julien Bouteloup for Flash Boys, 2020). The projects contract addresses found in the data set labeled `ArbitrageDAO` have been assigned to arbitrage.

**Arbitrage\_attack** Through various reports of flashloan attacks (Auguste (2020), Certik (2020*b*), PeckShield (2020), Trustnodes (2020*a*)), several addresses could be identified which could achieve an arbitrage profit by exploiting vulnerabilities in the SCs of the protocols. The respective owner addresses were manually checked for other SCs and, if a similar scheme was available, they were assigned to the same application.

---

<sup>31</sup>e.g.: 0x475adbbc7226b121077e4bae696c526a83750c55eed408a391c5a951b0d5c2b4

**Arbitrage\_hammerbots** On the famous “Black Friday” on March 12. and 13. 2020, a collapse of the price of ETH caused many CDPs to be undercollateralized. Keeper bots are used by DEX and are in charge to counter act against zero bids (bids for liquidations around zero). Block-native (2020) find evidence that a group of hammer bots managed to spam nodes with txns and hindering keeper bots to successfully place bids and keep zero bids from happening. 22 addresses of potential hammer bots have been identified in the report. These addresses could be identified within the data set of the present work. It is noticeable that these txn do not include any transfers other than taking out a flashloan and immediate repaying them. Hammer bot-like txns are identified by the mentioned addresses and when there are only two transfers involved in dYdX Flash Loans following the same scheme. Hence they do not seem to have any other purpose than to consume node resources. Additionally, it has been found that potential hammer bot addresses<sup>32</sup> have interacted with contracts which are assigned to Arbitrage\_liquidation (see 4.2 Arbitrage\_Liquidation). Because these addresses are only potential hammer bots it is not yet clear to which extend the found relation holds, but it seems that the strategy is comparable to actions observed on black friday. Although the purpose of these txns could not be fully clarified as a deeper analysis would exceed the scope of the work, it seems reasonable to assume that they serve as a preparation for arbitrage trades, as was the case on Black Friday.

Txns which are initiated by addresses that have been identified as potential hammer bots, because they also initiated txns which follow the same back-and-forth borrow/repay scheme, have been assigned to Arbitrage\_hammerbots in “subcat2” when another application have been found previously. Again it has to be stated that one should take care in interpreting these relationships and that a deeper analysis must be done beforehand.

---

<sup>32</sup>e.g. [txn of address: 0x03ebd0748aa4d1457cf479cce56309641e0a98f5 on Etherscan](#)

**Arbitrage protocols** The following protocols are known to be protocols with different arbitrage strategies:

- **Arb\_Balancer**: Balancer Offers the possibility for arbitrage traders to rebalance crypto portfolios. Fees for rebalancing are forwarded to portfolio owners (Balancer, 2019). Rebalancing of portfolios work similar as described for Uniswap (see: 3.4.1 FlashSwaps) with the difference that it is not limited to token pairs but can contain multiple assets. The strategy behind is to take out a flashloan and being able to rebalance large amounts of portfolios or included tokens when they are off market price and keep arbitrage profits. Txns including addresses of Balancer swaps have been assigned here.
- **Arbitrage\_Arbcontract**: The contract of Orfeed by Proofsuite enables flashloan based arbitrage trading. “Triangular arbitrage enables a user to perform a multi-point exchange of funds between specified Assets on supported decentralized exchanges” (mikedeshazer for Orfeed on Github, 2020). All txns interacting with the **Arbcontract** have been taken into consideration here.

### **Collateral\_swap**

**Collateral\_Swap\_0** By labeling known contract addresses, it was possible to identify txns interacting with addresses of the project Collateralswap directly.

### **CCDP**

**CCDP\_Defisaver** All txns interacting with the project Defisaver have been applied to the application CCDP directly (see: 2.2.1 Closing CDP). These include all addresses including the name “Defisaver” or the contracts **LoanShifterReceiver**, **CompoundSaverFlashLoan**, **MCDCloseFlashLoan**, **MCDFlashLoanTaker**,



MCDOpenFlashLoan, MCDOpenProxyActions, MCDSaverFlashLoan and CompoundImportFlashLoan<sup>33</sup>

## Wash trading

**Wash\_trading\_potential** It remains mostly unclear under which conditions a back-and-forth trade should be considered to be wash trading because back-and-forth trades may happen in ordinary swaps. Following the example in 2.2.1 Wash trading, all txns which do not contain any other transfers besides the flashloan, the back-and-forth transfer and a maximum of one other transfer to another address, were included. Furthermore, the back-and-forth transfer of the same amount in the same currency should take place to distinguish the txns from arbitrage txns. Furthermore, txns by potential hammer bots (see: Arbitrage\_hammerbots) were excluded as they follow the same scheme but cannot be classified as wash trading. Here it must be made clear that the identified txns must be treated as potential and not definitive wash trading txns, as no conclusive evidence of a volume manipulating intention can be established.

## No\_application

**No\_app\_Error** Txns in which an error occurred due to a revert, running out of gas or a bad instruction.

**No\_app\_Flash\_minting** Flash minting is not to be considered a specific application because here, all applications already described are conceivable. Nevertheless, the observed txns seem to have been used for test purposes only because there rarely transfers other than minting and burning observable. Therefore it cannot be assigned to a specific use case.

---

<sup>33</sup>contracts: [https://github.com/DecenterApps/\[...\]/contracts](https://github.com/DecenterApps/[...]/contracts)

## **NAPP**

No application could be assigned due to low occurrence, unknown scheme or because manual checks of all txns would not have been meaningful in view of the gain in additional explanatory potential.

### 4.3 Findings

The final data set consists of 6857 txns and 64660 token transfers. Table 1 provides an overview over the attributes used in the data set "Master-data.xlsx". For calculating USD values of tokens, the prices of the most frequent tokens used were collected. This was done because unfortunately exchange rates were to be found to infrequently available for most of the unfrequent tokens involved in the data set. This is especially true for the large amount of tokens found in FlashSwaps. This approach resulted in a coverage of almost 92% of all token prices, which is considered acceptable for a representative overview of the volumes. Exchange rates have been taken from Coinmarketcap (CoinMarketCap (2020a), CoinMarketCap (2020b), CoinMarketCap (2020c)). The relevant data can be found in the data set "Exchange\_rates.csv". WETH wurde durch ETH substituiert. [PASS DAS AN][PDF SUSI REVIEW]

Table 1: Attributes in final data set

Attribute	Description
uid	unique identification index per txn
transactionHash	txn hash
blockNumber	blocknumber of txn
protocol	protocol of initial flashloan in lowercase
timeStamp	timestamp of the block
fl	identification of dYdX Flash Loans
errDescription	description when error occurred
isError	error check
fname	t1 labeled
tname	t2 labeled
t1	transfer from
t2	transfer to
data	value of token transfer
tokenSymbol	Symbol of token transferred
tokenName	Name of token in transferred
tokenDecimal	decimal of token
application	application per txn
subcat	subcat per txn

The headers in the "Exchange\_rates.csv" data set correspond to the description provided above except the following: "curr" is the currency of the initial Flash Loan, Flashswap or flashminting. "timeStamp" is set to "DD-MM-YYYY" for better mapping of exchange rates. "val" is the value of the loan. "date" was used for mapping exchange rates (the same as "timeStamp"). "exchange\_rate" is the rate of conversion from "val" to "usd" which is the value of the loan in USD.

The following section shows the distribution of txns over protocols and its distribution over time on protocols. To find reasoning for this, it shows which applications and subcategories have been used the most and in which months they could be observed. It then presents the usage of tokens per protocol and concludes by showing USD volumes of Flash Loans over all, on protocols and for applications and subcategories.

If possible, an interpretation of the results is given. However, it is important to note that the focus has been deliberately placed on a qualitative analysis. Although the theory-building assumptions for the explanation of observations are supported by the data basis with descriptive statistics, to a large extent no significance analyses or regression models are provided. This is therefore appropriate for the present work, which aims to explain the hitherto very unilluminated topic of Flash Loans and to give a description of the applications and the monetary use to date. All visualizations of data in figures have been done by using Seaborn of Waskom (2012-2020) in Python.

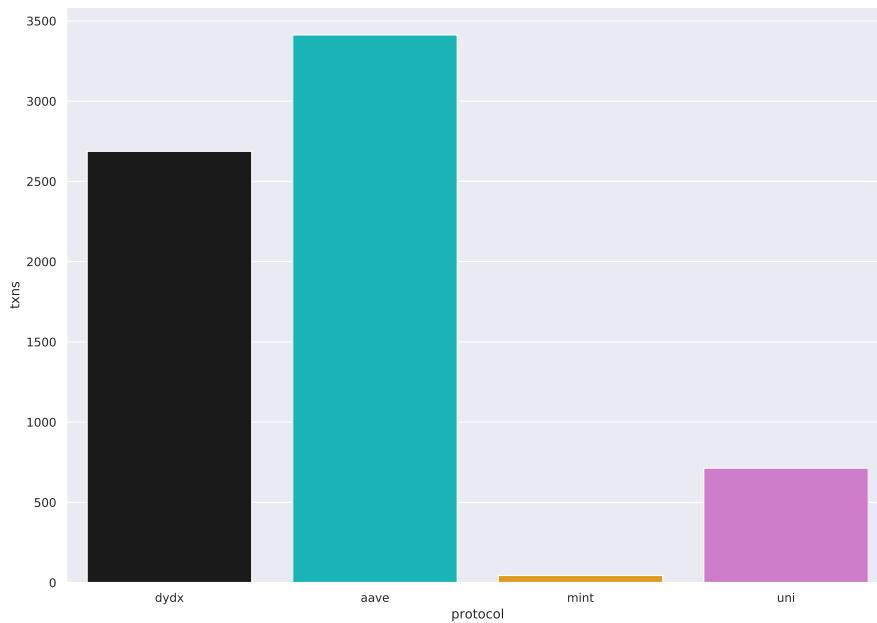
#### **4.3.1 Utilization numbers of Flash Loans**

It was found that Aave covers half of all Flash Loans taken out in the observed time frame. This is attributable to the relatively early popularity of the implicit flashloan feature in January 2020 (Frangella, 2020). dYdX covers a share of 39.19%. It will be shown later that dYdX is mainly used for arbitrage which is considered to be caused by lower fees. Aave is mainly used for CCDP because Defisaver makes use of Aave Flash Loans and is therefore only used here. Uniswap only covers a small amount

(10.40%) which is due to the possibility of flash swaps became available only in May 2020. Further it can be observed that the concept of Flash minting is hardly ever used. This is due to the very new and unknown concept which is almost not covered in any reports. On bZx, the only observable flashloan is the second flashloan attack described in 3.5 Flash loan attacks. No definitive statement can be made here on the low level of use. However, it is assumed that the hardly available documentation, apart from audit reports (see: CertiK (2020a)) in conjunction with the paused protocol following the attack (Khatri, 2020), which is expected to relaunch on August 31 (CoinMarketCal, 2020), is the reason for no further calls of the function. Table 2 and Figure 5 visualize the distribution of txns and transfers over protocols. Flash loans on Aave and dYdX include around 10 transfers per Flash Loan. Uniswap only includes an average of 3.31 transfers per txn. This corresponds to a lot of failed txns on Uniswap as shown later. The same is true for Flashminting. The average transfers on bZx is high because there is only one txn observable.

Table 2: Distribution of Flash Loans over protocols

Protocol	txns	%	transfers	%	$\emptyset$ transfers
Aave	3412	49.76%	34558	53.45	10.12
dYdX	2687	39.19%	27703	42.84	10.31
Uniswap	713	10.40%	2363	3.65	3.31
Flashminting	44	0.64%	36	0.06	0.81
bZx	1	0.01%	47	0.07	47
Total	6857		64660		



**Figure 5:** Distribution of flashloan txns over protocols  
Source: own representation

### 4.3.2 Distribution of txns over time

The following part describes the distribution of txns over time on protocols and tries to reason the observed occurrences by showing the distribution of subcategories over time on protocols.

Table 3 show the monthly usage of flashloan txns. While numbers started to rise since the launch of the feature on Aave in February, the usage is still very moderate.

Table 3: monthly usage of Flash Loans

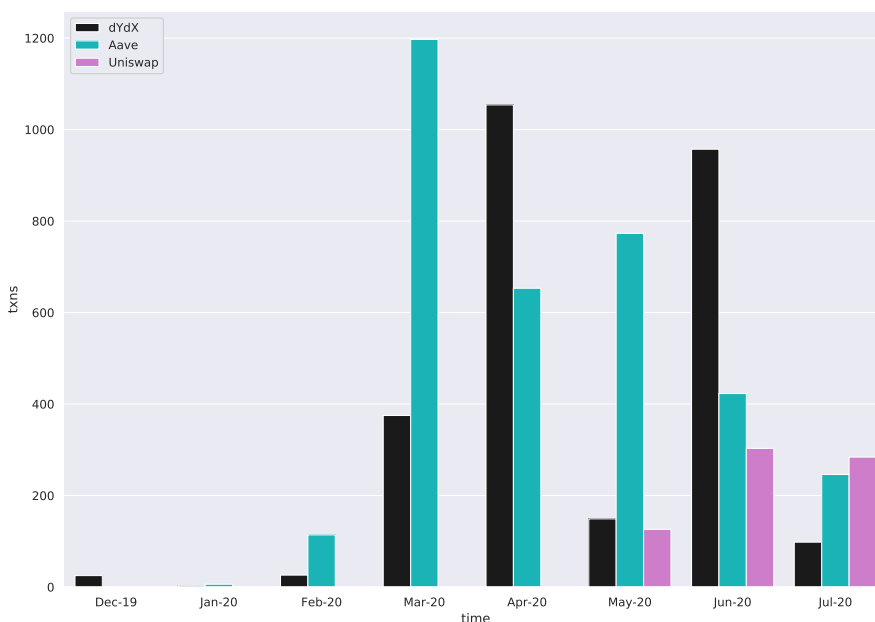
Month	txns
Dec-19	25
Jan-20	9
Feb-20	157
Mar-20	1574
Apr-20	1726
May-20	1055
Jun-20	1683
Jul-20	628
Total	6857

The average number of txns per month from February to June 2020 is 1509.5 which shows the very moderate use. December 2019 and January 2020 were excluded in calculating the average because Flash Loans were officially only available from February on. July was excluded because there is no data available for the whole month.

Figure 6 and table 4 show the distribution of number of txns on the different protocols on a monthly base and supports the hypothesis that the observed differences in the quantity of Flash Loans executed are partly related to the time when the flashloan feature of Aave and Uniswap became widely known. Uniswap V2 was launched in May 2020 which explains observable FlashSwaps only starting in that month with a relatively low average share of  $\sim 21\%$  in May, June and July. It has to be noted that data for July is scarce, since data collection was done until 15.07.20.

Table 4: txn count over time

Month	dYdX	Aave	Uniswap
December-19	25	0	0
January-20	3	6	0
February-20	26	114	0
March-20	375	1197	0
April-20	1054	653	0
May-20	149	773	126
June-20	957	423	303
July-20	98	246	284
Total	2687	3412	713



**Figure 6:** Distribution of txns on protocols over time

Source: own representation

### 4.3.3 Applications

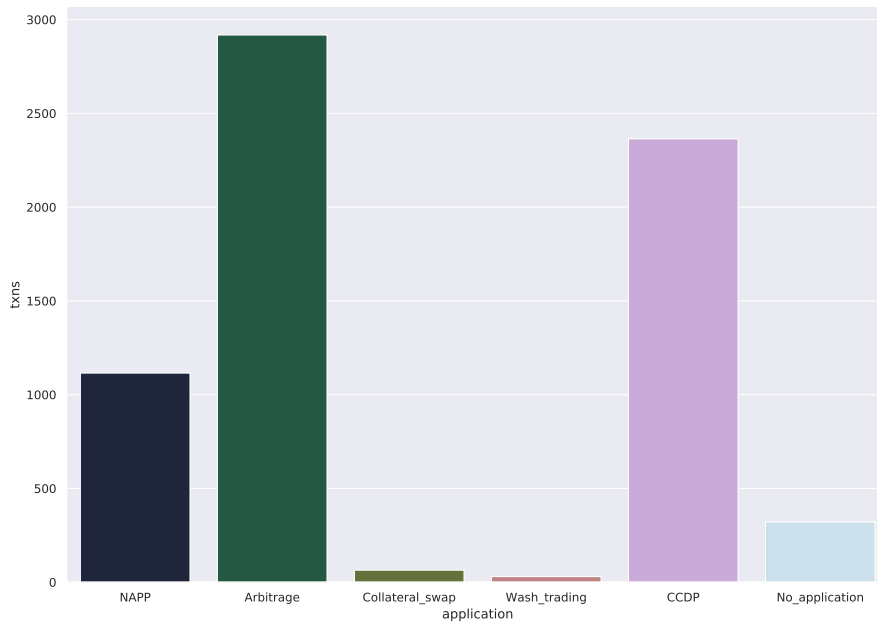
The following part illustrates the distribution of applications over all protocols and explains the observations by showing the individual shares of subcategories on a certain protocol. Over the whole data set, 5724 txns



could be assigned to an application, which corresponds to  $\sim 83\%$ . Therefore 1151 txns remain unassigned (NAPP). Table 5 and figure 7 provide an overview of the distribution of applications over flashloan txns. It is found that the most frequent use cases are Arbitrage (43.21%) and CCDP (35.09%). No\_application was assigned to txns which showed an error because the initial purpose of said txns could not be determined. All these txns were observed on Uniswap. Being aware that Uniswap only have been requested for Flash swaps with the purpose of different arbitrage strategies, it seems reasonable that these txns may have followed a similar purpose. Although it must be noted that a large share of Uniswap txns ( $\sim 22\%$ ) could not be assigned to an application at all. Collateral swaps only make up a share of under 1% of all txn applications. There are only 30 txns which could be identified as potential wash trading. This is mainly due to the loose definition of back-and-forth trades having a wash trading purpose or not. bZx and flash minting were excluded here since they are not representative for the analysis of the applications. Therefore the data set for the following analysis reduces to 6812 txns.

Table 5: Distribution of applications over flashloan txns

application	txns	%
Arbitrage	2918	42.83
CCDP	2364	34.70
NAPP	1115	16.36
No_application	321	4.71
Collateral_swap	64	0.94
Wash_trading	30	0.44
Total	6812	



**Figure 7:** Distribution of txns over applications  
Source: own representation

Table 6 shows the number of unique occurrences of subcategories in txns on the three protocols. The column “% protocol” corresponds to the occurrence of a subcategory on a protocol relative to all other subcategories on that protocol. “% all” shows the share of a subcategory relative to all other subcategories on all protocols.

Table 6: Distribution of subcategories over protocols

txns	subcategory	application	protocol	% protocol	% all
2365	CCDP_Defisaver	CCDP	Aave	69.31	34.72
931	Arbitrage_bot	Arbitrage	dYdX	34.65	13.67
781	Arbitrage_hammerbot	Arbitrage	dYdX	29.07	11.47
567	NAPP	NAPP	dYdX	21.10	8.32
474	Arbitrage_bot	Arbitrage	Aave	13.89	6.96
385	NAPP	NAPP	Aave	11.28	5.65
334	Arbitrage_swap	Arbitrage	dYdX	12.43	4.90
326	No_app_error	No_application	Uniswap	45.72	4.78
228	Identified_Arbitrage_HXY	Arbitrage	Uniswap	31.98	3.35
158	NAPP	NAPP	Uniswap	22.15	2.31
64	Collateral_swap_0	Collateral_swap	Aave	1.88	0.94
41	Arbitrage_Liquidation	Arbitrage	Aave	1.20	0.60
38	Arbitrage_swap	Arbitrage	Aave	1.11	0.56
25	Arbitrage_Liquidation	Arbitrage	dYdX	0.93	0.37
24	Wash_trading_potential	Wash_trading	dYdX	0.89	0.35
17	Arbitrage_hammerbot	Arbitrage	Aave	0.50	0.25
12	Arbitrage_Arbcontract	Arbitrage	Aave	0.35	0.18
10	Arb_Balancer	Arbitrage	dYdX	0.37	0.15
6	Wash_trading_potential	Wash_trading	Aave	0.18	0.09
6	Arbitrage_dao	Arbitrage	Aave	0.18	0.09
6	Arbitrage_Arbcontract	Arbitrage	dYdX	0.22	0.09
5	Arbitrage_comp	Arbitrage	dYdX	0.19	0.07
4	Arbitrage_attack	Arbitrage	dYdX	0.15	0.06
4	Arb_Balancer	Arbitrage	Aave	0.12	0.06
1	Arbitrage_comp	Arbitrage	Uniswap	0.14	0.01
6812					

CCDP\_Defisaver being the most used unique subcategory makes up 34.72% of all txns and almost 70% of txns on Aave. As said before, Aave is mainly used for CCDP because Defisaver is making use of Aave Flash Loans. Nearly all other identified txns on Aave are initiated by Arbitrage bots (13.89%). dYdX show a high occurrence of Arbitrage bots and potential hammer bots, making up almost 65% of the protocol usage and one fourth of all txns. Further, a lot of Arbitrage swaps (12.43%) can be observed. The split up in usage between Aave and dYdX may be explained by the difference in fees for taking up Flash Loans. When executing arbitrage strategies it seems reasonable to make usage of the most cheap protocol to to achieve the highest possible profit. Whereas the comfortable use of Defisaver for CCDP seems to justify paying a higher fee for keeping collaterals from being liquidated. This seems also to be

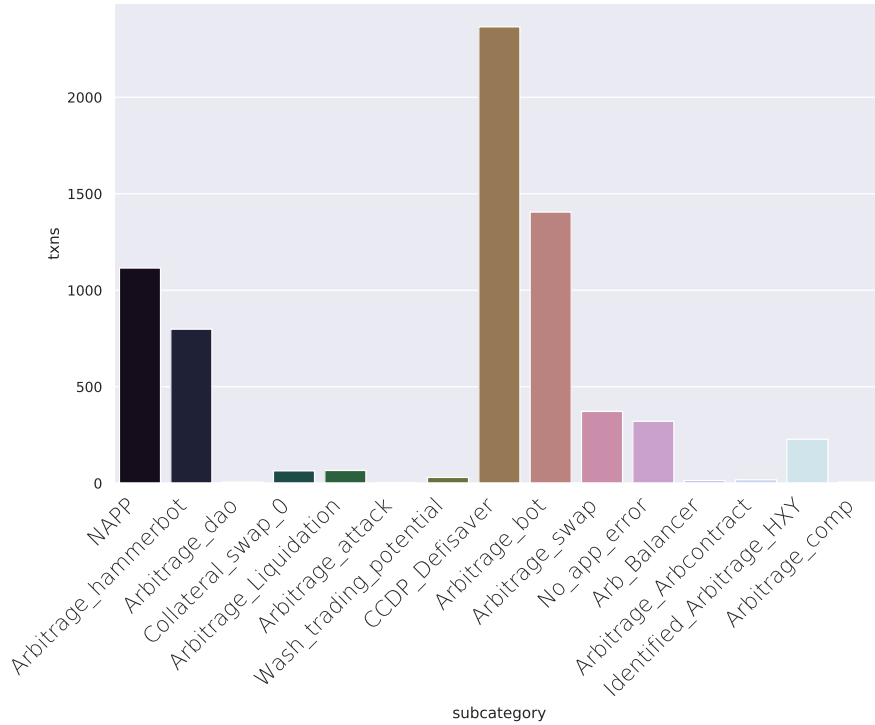
true for Collateral swaps whose only being executed using Aave as well. Uniswap show high share of 45.72% of failed txns. Table 7 shows that most errors have been caused by a revert. It remains yet unclear if the cause of txns beeing reverted have been an insufficient repayment of the token pair or if another requirement have not been fulfilled.

Table 7: Txns error of failed txns on Uniswap

errDescription	txns	%
Reverted	302	92.63
Out_of_gas	8	2.45
Bad_instruction	16	4.90
Total	326	

Other txns happening on Uniswap whose application could be identified are using interest arbitrage on HEX.win (31.98%) because Uniswap offers easy access to HEX for executing the arbitrage strategy.

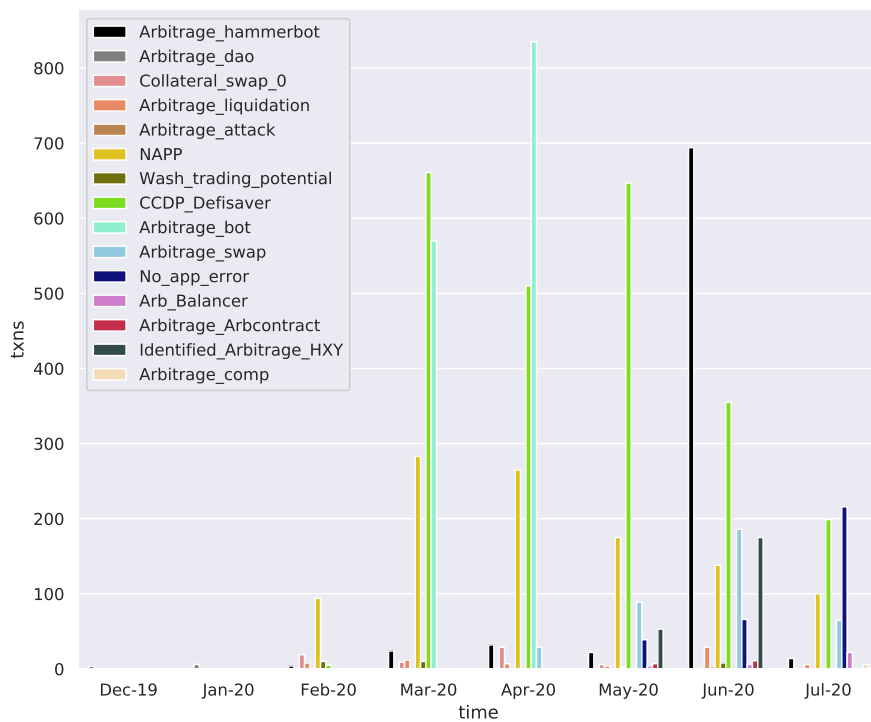
Figure 8 shows the occurrence of all subcategories. It can be observed that CCDP\_Defisaver, arbitrage bots and potential hammer bots make up the major share of identified subcategories.



**Figure 8:** Distribution of txns over applications subcategories  
Source: own representation

Figure 9 illustrates the usage of subcategories over the time horizon of the data set. No significant flashloan txn is observed in December and January. The only ones occurring are on dYdX. It is assumed that these were not carried out with the intention of initiating a flashloan, as the concept of Flash Loans was not known at that time, but it is worth noting that it was already possible to borrow and repay the same amount within one txn on dYdX. Txns on February 2020 consist of a large share of unknown purpose. The highest usage of Flash Loans in March can be found in CCDP\_Defisaver (~ 42%). It was found that CCDP\_Defisaver txns are noticeable high for following dates: On March 12 2020, the so called “Black Thursday”, the price of ETH dropped by around 30% in 24 hours (PULSE, 2020), leading to “black friday” one day later. Many tokens followed the downward trend, causing users’ collaterals to drop in value and CDPs to be in danger of liquidation (see also: 4.2 Ar-

bitrage\_hammerbots). A comparable price drop happened on May 10 ( $-10.38\%$ )(Cointelegraph, 2020).



**Figure 9:** Distribution of applications on all protocols over time  
Source: own representation

Table 8: Top three CCDP txns per day

txns	date
132	Mar 12, 2020
92	Mar 13, 2020
91	May 10, 2020

Table 8 shows the three most frequent occurrences of CCDP txns per day. It is noticeable that the presented txns happen on dates when great price drops of ETH could be observed. Nevertheless, no strong evidence for a general relation between the price of ETH and the number of CCDP txns could be found. Figure 10 shows the relationship between the price

of ETH (taken from investing.com (2020)) and observed CCDP txns by a linear regression line on a 95% confidence interval. The light blue bands around the line represent the standard error of the regression line. It illustrates that CCDP txns can not be observed relatively often when prices tend to be relatively high. There is a small observable relationship between the price of ETH and the usage of CCDP, but standard errors are wider spread for high occurrences. Three points on the left-hand side stick out the most (marked in red) which are related to the three dates mentioned beforehand. The three outliers seem to raise the regression line on the left, which is partly visible in the increase in standard errors.

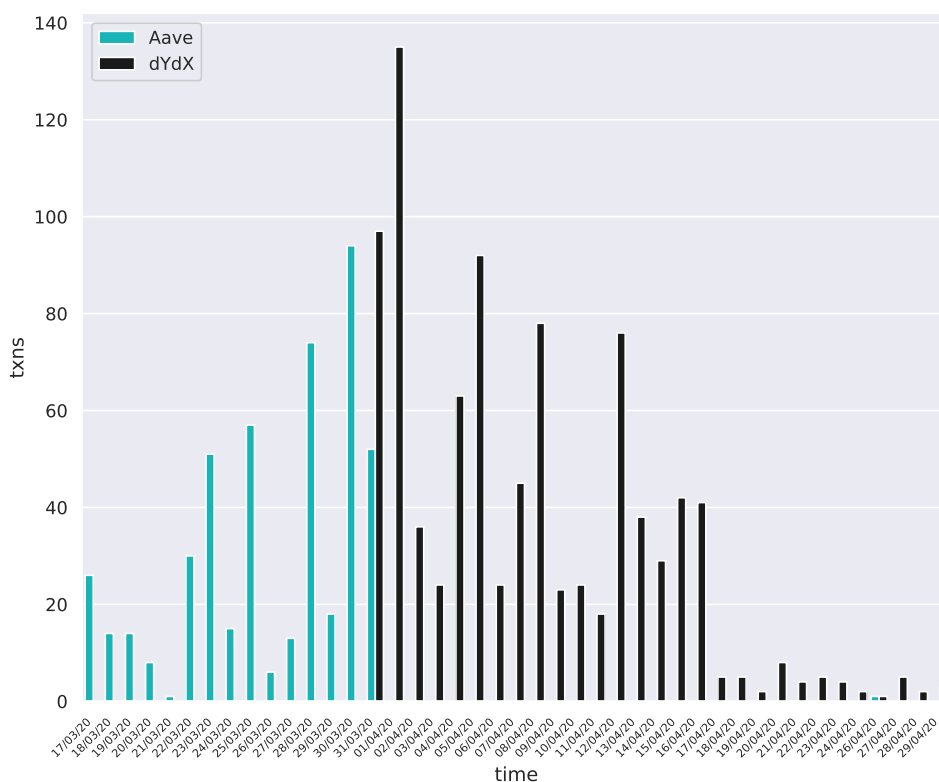


**Figure 10:** Linear regression CCDP txns and price of ETH  
Source: own representation

Although no general relationship between the usage of Defisaver and the

occurrence of price drops could be determined, it seems reasonable to say that specific dates of price drops caused users to try to save collaterals from getting liquidated. The share of nearly 70% of all txns by CCDP on Aave tend to explain observable high volumes in March, April and June observed from figure 6.

A large proportion ( $\sim 42\%$ ) of txns in March and April are arbitrage bots. First performed exclusively on Aave, the bots change to almost exclusively been executed on dYdX from April 1, 2020 (with one exception on April 24, 2020). Figure 11 shows this change by displaying the distribution of arbitrage bot txns over time on protocols on a daily base. The large arbitrage bot txns in April of about 79% on dYdX explains the relatively high increase in txn observed on dYdX in figure 6.



**Figure 11:** Distribution of arbitrage bot txns over time on protocols  
Source: own representation



A noticeable large amount of potential hammer bot txns happens in June 2020. Table 9 presents the five most observations per day of potential hammer bot txns. It shows a tight concentration of txns in the time between June 22 and June 27. Despite the observation of frequent common appearance, an interpretation is still difficult. It is yet not clear what purpose the transaction serves, but it seems quite possible that by consuming node resources they are doing preparation for further arbitrage strategies. However, as stated in 4.2 Arbitrage\_hammerbots the data basis for a conclusion is missing. The high share of txns on dYdX by named addresses of around 73% can be observed in the high volume of txns on dYdX in June back in figure 6.

Table 9: Top five potential hammer bot txns per day

txns	date
512	Jun 22, 2020
81	Jun 27, 2020
34	Jun 23, 2020
28	Jun 24, 2020
10	Jun 11, 2020

#### 4.4 Token on protocols

In the following it is described which tokens are used on the different protocols and what share they make up.

**Aave** Table 10 shows the distribution over tokens of Aave Flash Loans, the number of txns a Flash Loan in a specific token was taken out, the percentage share over all txns and the average amount of one Flash Loan of a token. Again it must be noted that the conversion rate of the tokens to USD is based on a snapshot from Etherscan on 31.07.2020. Therefore DAI (55.10%), ETH (34.61%) and USDC (6.89%) are most frequently used.

Table 10: Flash Loan amounts and distribution of tokens on Aave

Symbol	Txns	%	Amount	∅ Amount
DAI	1880	55.10	43777101.86055	23285.69248
ETH	1181	34.61	22262.50215	18.85055
USDC	235	6.89	524860.59799	2233.44935
BAT	70	2.05	293092.74916	4187.03927
WBTC	24	0.70	4.23861	0.17661
ZRX	14	0.4	3246.67770	231.90555
USDT	5	0.15	48725.43332	9745.08666
REP	1	0.03	0.67277	0.67277
LINK	1	0.03	617.44845	617.44845
MKR	1	0.03	5.00000	5.00000
Total	3412			

**dYdX** Table 11 shows the distribution over tokens of dYdX Flash Loans. ETH (42.24%), DAI (36.70%) and USDC (21.06%) are exclusively used. Without taking into account USD values of the observed token transfers, it can already be observed that dYdX volumes are well above those on Aave. DAI Flash Loans on dYdX are about 30%, ETH about 17220% and USDC about 756% above Flash Loans on Aave. Although there are less Flash Loans taken out on dYdX, the average amount is higher in general. This seems to be reasonable when taking into account that dYdX is mainly used for arbitrage purposes and Aave for CCDP.

Table 11: Flash Loan amounts and distribution of tokens on dYdX

Symbol	Txns	%	Amount	∅ Amount
DAI	986	36.70	57121686.9440938	57932.7453794054
ETH	1135	42.24	3856074.42108517	3397.42239743187
USDC	566	21.06	4491627.476891	7935.73759168021
Total	2687			

**Uniswap** Table 12 shows the distribution over tokens of Uniswap FlashSwaps. HEX (33.52%) and WETH (6.45%) are used the most.

Table 12: Flash Loan amounts and distribution of tokens on Uniswap

Symbol	Txns	%	Amount	∅ Amount
iserr	0	0.00	0	0
HEX	239	33.52	12520186.5674295	52385.7178553535
WETH	46	6.45	274.629507614833	5.97020668727898
SLP	16	2.24	25048	1565.5
UBT	8	1.12	5762.74387153	720.34298394125
DAI	7	0.98	5000.00102243845	714.28586034835
BAT	5	0.70	33944.8159269802	6788.96318539603
COMP	5	0.70	1053.171	210.6342
ISLA	4	0.56	1047.76218132451	261.940545331128
XIO	4	0.56	16980.0176925845	4245.00442314613
KNC	3	0.42	2144.47440080784	714.82480026928
SNX	3	0.42	1529.51386040405	509.837953468017
AMPL	2	0.28	4645.82576914	2322.91288457
AUC	2	0.28	1866.2566711331	933.128335566552
BUIDL	2	0.28	406.306027279666	203.153013639833
CEL	2	0.28	555.8335	277.91675
DATA	2	0.28	13212.3511596386	6606.1755798193
ESH	2	0.28	330.044992105303	165.022496052652
JRT	2	0.28	5397.17218807722	2698.58609403861
LINK	2	0.28	747.11708571948	373.55854285974
MKR	2	0.28	5.76443003453046	2.88221501726523
NEXO	2	0.28	1998.8867129342	999.443356467098
SHIP	2	0.28	5831.73432078942	2915.86716039471
USDC	2	0.28	14871.492006	7435.746003
ANT	1	0.14	403.465023090514	403.465023090514
BAND	1	0.14	98.6562935008362	98.6562935008362
BLT	1	0.14	1612.19043858256	1612.19043858256
BNT	1	0.14	224.655563758628	224.655563758628
DONUT	1	0.14	3639.0268454507	3639.0268454507
DZAR	1	0.14	20522.392116	20522.392116
GRID	1	0.14	175.504960223652	175.504960223652
GST2	1	0.14	8.61	8.61
LRC	1	0.14	363.717615226304	363.717615226304
MCX	1	0.14	7941.32268774983	7941.32268774983
MINDS	1	0.14	106.324926180489	106.324926180489
OXT	1	0.14	301.403861505878	301.403861505878
QNT	1	0.14	2.8749153140519	2.8749153140519
REN	1	0.14	1976.90332630488	1976.90332630488
RLC	1	0.14	607.287036635	607.287036635
RPL	1	0.14	894.24341207903	894.24341207903
STAC	1	0.14	57267.4033322954	57267.4033322954
STAKE	1	0.14	2.37401091799079	2.37401091799079
UP	1	0.14	5822.64377510077	5822.64377510077
VXV	1	0.14	38.5603547551341	38.5603547551341
WBTC	1	0.14	0.23748504	0.23748504
ZRX	1	0.14	146.130895378452	146.130895378452
cDAI	1	0.14	2751.0697203	2751.0697203
Total	3412			

A large share of 45.72% is failed transactions which show no value at

all (“iserr”). The many different tokens is made up by the fact that FlashSwaps allow the swapping of any token pair available. This makes it hardly comparable to Aave and dYdX when it comes to the volumes of certain tokens. The stated amounts from the token which was given to the user in a txn has been used, not the token which was paid back.

**Flash minting** The analysis of the 44 flash minting transactions showed hardly any relevant results for the task of this study. Around 18% of txns failed (4 out of gas and 4 reverted). A sum of 11.300442018246 fmETH and 11.122 fWETH were used in txns. Only one txn showed more than one transfer. No applications could be identified either. This is because the concept is rarely used and the present txns seem to have been initiated for test and proof-of-concept purposes only<sup>34</sup>. The conclusion is based on the fact that almost no transfers other than the minting itself can be observed and the amounts are very low. Although there are concepts in the making which try to use the concept, such as flash-mint.app for the easy use of flashminting<sup>35</sup>, a new token ADEX with inherent flash minting feature<sup>36</sup> or a request to make DAI on MakerDAO flash mintable<sup>37</sup>, these concepts do not show observable data for the period of time under consideration.

## 4.5 USD volumes

The following presents the volume of Flash Loans taken out in USD over all and over specific protocols and applications. From the previous analysis, it could be concluded that DAI, ETH and UDC make up a share of 91.88% of all tokens used. As a result, only historical prices for this tokens were acquired since the expense of collecting prices of other tokens were not to be found appropriate.

Table 14 and figure 12 show the value of Flash Loans taken out per

---

<sup>34</sup>According to Austin Williams in a personal discussion via Mail 11.08.2020

<sup>35</sup><https://flashmint.app>

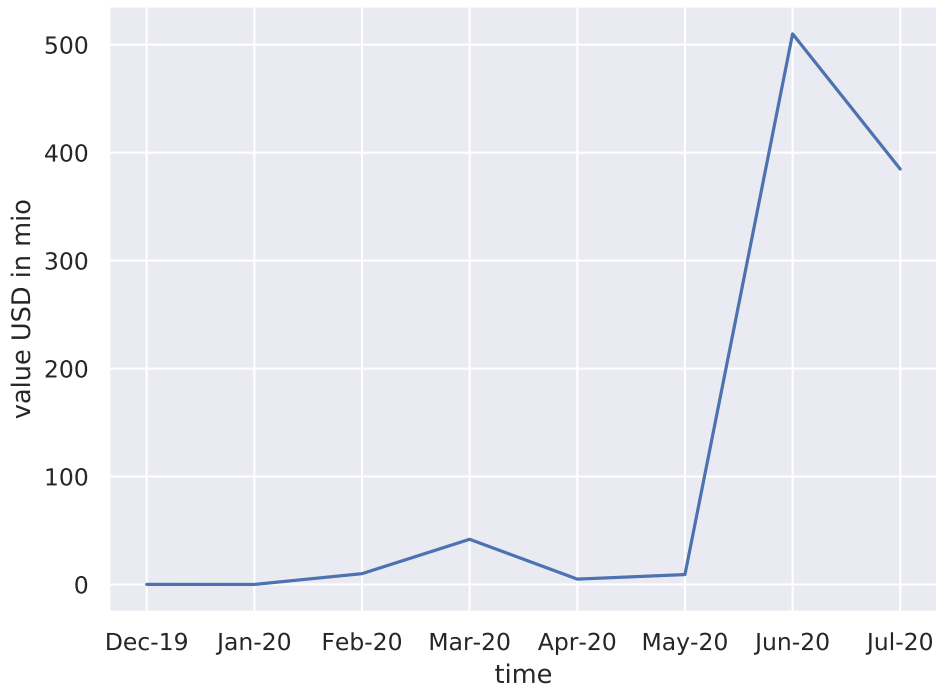
<sup>36</sup><https://www.adex.network/blog/token-upgrade-defi-features>

<sup>37</sup>[https://forum.makerdao.com/\[...\]-flash-mint-module/3635](https://forum.makerdao.com/[...]-flash-mint-module/3635)

month in mio USD. After relatively low volumes in December 2019 until February 2020, volumes raise quickly im March when Flash Loans became more popular and Black Thursday and Friday made it necessary for users to save collaterals through Defisaver. After relatively low volumes compared to the following months, June and July show high USD volumes of Flash Loans.

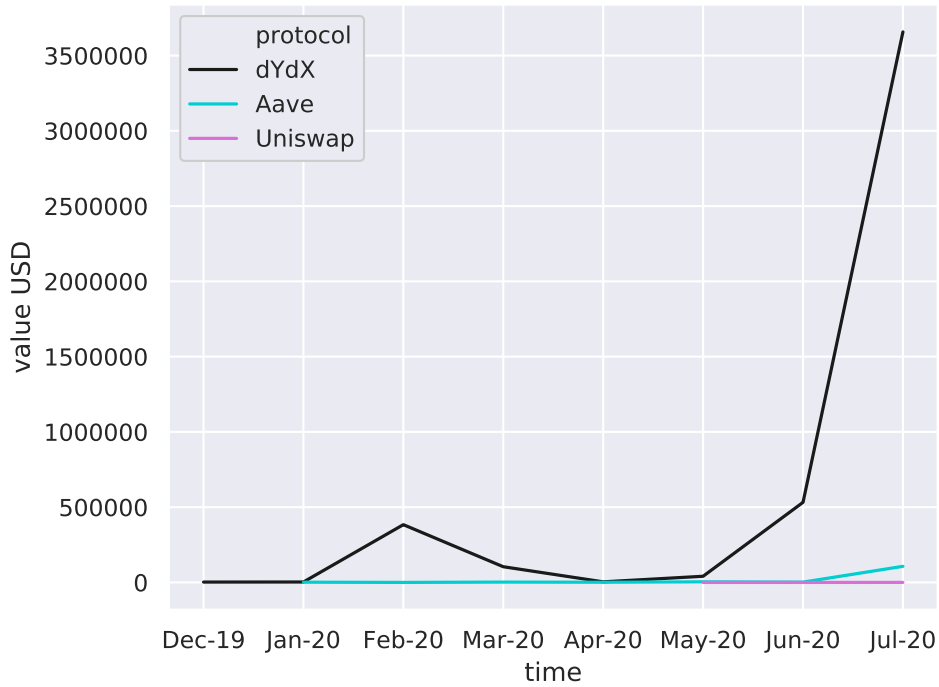
Table 13: Flash Loan value in mio USD per month

month	value in mio USD
Dec-19	0.060188488682491
Jan-20	0.018009891791302
Feb-20	9.99398983251672
Mar-20	41.8248923026221
Apr-20	4.94490782522818
May-20	9.1571141102114
Jun-20	510.02652879264
Jul-20	384.753773424701
Total	960.779404668394



**Figure 12:** Flash Loan value in mio USD per month  
Source: own representation

Figure 13 illustrate the volume of Flash Loans in USD taken out on protocols by month. It is quite clear that dYdX shows by far the most volume of all three major platforms, whereas Aave remains relatively flat in comparison. It should be noted that the results for Uniswap are not very meaningful as the available USD values of the tokens are not representative of the use of tokens in FlashSwaps.



**Figure 13:** USD volume of Flash Loans on protocols per month  
Source: own representation

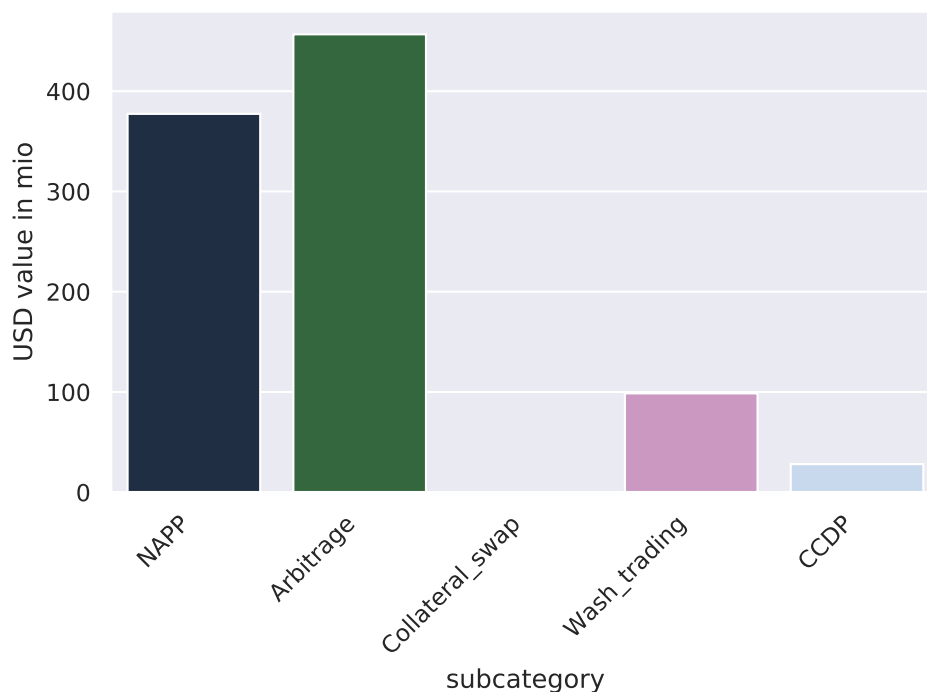
Table 14 shows the aggregated Flash Loan values in USD on dYdX and Aave. Looking at the use of the platforms in relation to their application, the results appear to be consistent. Since dYdX is mainly used for arbitrage, the higher values seem to make sense

Table 14: Flash Loan value in USD per protocol

protocol	value in USD
dydx	927,014,127.795046
aave	33,745,408.0583357
total	960,759,535.853382

The volume in mio USD of Flash Loans taken out for a specific application is shown in figure 14. It supports the assumption that arbitrage accounts for the majority of observed USD values. Another large part is

made up of undefined applications. Potential washtrading txns, CCDP and collateral swap, on the other hand, do not show high values.



**Figure 14:** Volume of Flash Loans for applications in USD  
Source: own representation

The volume in USD of Flash Loans taken out for a specific applications subcategory is shown in figure 15.

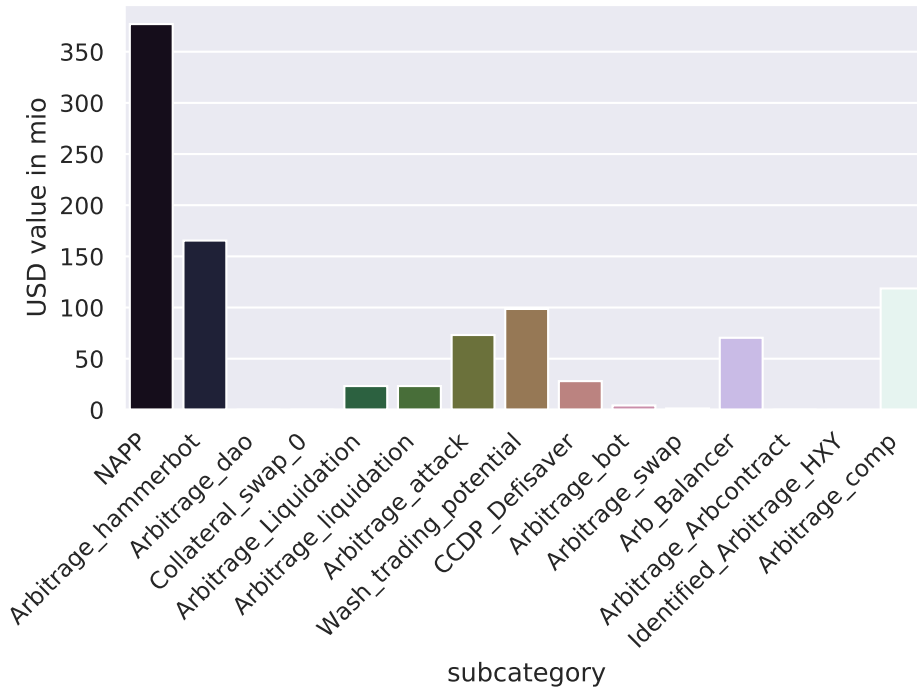
Unidentified applications show the largest share followed by potential hammer bots. The relatively high proportion of the latter might be due to the very large number. Surprisingly, arbitrage bots do not account for a large share of the USD amounts executed. Arbitrage\_comp stands out because it is strongly influenced by a txn where WETH (substituted by ETH in the data set) worth more than 42 million USD was borrowed on dYdX in a Flash Loan<sup>38</sup>. This is also true for potential wash trading txns where a few outliers tend to drive up the bar (for the txn hashes see data

<sup>38</sup>txn:

0x63a99ac8b81ff2ee1c80bf9f0ca5809a48cfa54e88fc3f38b1cca94bec99536



set "Exchange\_rates.csv"). Also noticeable are arbitrage attacks which only show four txns but an average of 18,291,253.58 USD per txn. The same seems to be true for Arb\_Balancer, where the average value per Flash Loan in USD is 5,031,584.033



**Figure 15:** Volume of Flash Loans for applications subcategories in mio USD

Source: own representation

## 5 Conclusion

The aim of this paper was to explain the novel concept of Flash Loans and to provide an analysis of their use. The result is an overview of how Flash Loans work and the presentation of possible applications as well as a description of the famous attacks. A comprehensive and informative data set has been created and a descriptive analysis shows its content and, where possible, describes conclusions drawn. It has been shown that arbitrage is the most common application of Flash Loans, followed by CCDP in times of large price drops of cryptocurrencies. Other applications occupy only a small part. Potential wash trading txns make up a very small part of the total applications. This is not surprising as on-chain wash trading is easy to identify and known DEX would not be of much use due to potential reputational damage. Wash trading similar txns through unidentified addresses seems to exist through and through in the case of potential hammer bots. However, the final purpose of the txns must remain unanswered here. Due to the strong involvement in past mempool manipulations of the addresses, a classification as arbitrage for these back-and-forth transfers seems more appropriate than being wash trading txns. Concerning the division of applications into subcategories, it was shown that bots, arbitrage- or hammer bots, take an essential part within arbitrage strategies. Other applications are mainly arbitrage swaps and HXY arbitrage strategies. Concerning the use of the different protocols, Aave takes the majority of txns, closely followed by dYdX. Uniswap shows little share, probably due to the short availability of flash swaps. A different picture emerges when considering the amount of Flash Loans taken out in token and the respective USD amount. It became clear that dYdX shows by far the highest amounts. Aave shows only a small share and Uniswap showed too much different coins for an appropriate analysis. Flash minting is hardly used at the moment, and it remains to be seen how Flash Loans on bZx will develop once the protocol is live again. In the course of the description of the flashloan attacks it became clear that Flash Loans uncovered the weaknesses of SCs and the interaction of protocols with Oracle contracts. Furthermore

it could be observed in arbitrage strategies (Arbitrage\_HXY and Arbitrage\_comp) that it is now easier for users to take advantage of interest rate strategies because interactions with protocols and the generation of incentive tokens is possible without actually having to raise own funds. It has been made clear that Flash Loans offer a wide range of possible use cases and that the usage in sheer number of txns is not very high yet. Compared to moderate numbers of txns, the cumulated volume of Flash Loans in USD of 960.78 mio and the average volume per Flash Loan of around USD 141,042 show that users make use of high values for their purposes.

## 5.1 Limitations and further research

The present thesis is limited to the analysis of Flash Loans and similar concepts on the presented platforms and analyzes for this purpose the use in different applications. There is no detailed analysis of used SC for the individual use cases. Furthermore, the present work is lacking in-depth analyses of certain subcategories of applications. To check for clear evidence of mempool manipulation by hammer bots, a analysis of mempool data should be carried out. To get insights into potential wash trading strategies, the respective volumes of transferred tokens at the time following the txn should be checked to examine if there was an effect on trading volumes. The paper presents the current state of the art of Flash Loans within a fixed time frame. In an SC based environment like Ethereum a constant new and further development can be observed. Around the concept of Flash Loans the emergence of new platforms like the user friendly use of flash minting<sup>39</sup>, new token with inherent flash minting feature<sup>40</sup> or aggregation of different Flash Loan protocols<sup>41</sup> can be observed. In further steps the results could be extended continuously. It must also be noted that a large part of the time and effort involved in the present work was caused by data acquisition. Especially for the

---

<sup>39</sup><https://flashmint.app>

<sup>40</sup><https://www.adex.network/blog/token-upgrade-defi-features>

<sup>41</sup><https://www.kollateral.co>

identification of Flash Loans on dYdX, a time consuming data cleansing was necessary. Therefore only data up to July are available. Based on the findings of this work, data should be relatively much easier to obtain and the data set could be expanded. Additionally, no detailed analysis of failed txns was in scope of the present work. However, it might be useful to find out the reasons for the failure of so many txns on Uniswap in order to draw conclusions about the potential use cases.

## References

- Aave (2020*a*), ‘Aave Protocol Whitepaper V1.0’, <https://whitepaper.io/document/533/aave-whitepaper>. Accessed: 08.05.2020.
- Aave (2020*b*), ‘LendingPool.sol’, <https://github.com/aave/aave-protocol/blob/7998c8a5a5fc8c326b261832358b49fa3a9e8b93/contracts/lendingpool/LendingPool.sol#L889>. Accessed: 04.07.2020.
- Aave developers (2020*a*), ‘Performing a Flash loan’, <https://docs.aave.com/developers/tutorials/performing-a-flash-loan>. Accessed: 08.05.2020.
- Aave developers (2020*b*), ‘What is Aave’, <https://developers.aave.com/#retrieve-contract-instances>. Accessed: 08.05.2020.
- Auguste, K. (2020), ‘The bZx attacks explained’, <https://www.palkeo.com/en/projets/ethereum/bzx.html>. Accessed: 16.07.2020.
- Austin Williams (2020), ‘Flash-mintable Asset-backed Tokens’, <https://github.com/Austin-Williams/flash-mintable-tokens>. Accessed: 08.05.2020.
- Balancer (2019), ‘Whitepaper’, <https://balancer.finance/whitepaper>. Accessed: 01.08.2020.
- Blockchain Transparency Institute (2019), ‘BTI Market Surveillance Report – September 2019’, <https://www.bti.live/bti-september-2019-wash-trade-report>. Accessed: 08.05.2020.
- Blocknative (2020), ‘Evidence of Mempool Manipulation on Black Thursday: Hammerbots, Mempool Compression, and Spontaneous Stuck Transactions’, <https://blog.blocknative.com/blog/mempool-forensics>. Accessed: 29.07.2020.
- bZeroX, LLC (2018), ‘bZx Litepaper’, [https://bzx.network/pdfs/bZx\\_lite\\_paper.pdf](https://bzx.network/pdfs/bZx_lite_paper.pdf). Accessed: 20.07.2020.

- CertiK (2020a), ‘Audit Report’, <https://bzx.network/pdfs/CertiK%20Verification%20Report%20for%20bZx.pdf>. Accessed: 21.07.2020.
- Certik (2020b), ‘Little Pains, Great Gains: How the Balancer DeFi Contract Was Drained’, <https://certik.io/blog/technology/little-pains-great-gains-balancer-defi-contract-was-drained>. Accessed: 25.07.2020.
- CoinMarketCal (2020), ‘31 August 2020 (or earlier) Protocol Relaunch’, <https://coinmarketcal.com/en/event/protocol-relaunch-38288>. Accessed: 12.08.2020.
- CoinMarketCap (2020a), ‘Dai’, <https://coinmarketcap.com/currencies/multi-collateral-dai/historical-data/?start=20190820&end=20200820>. Accessed: 20.08.2020.
- CoinMarketCap (2020b), ‘Ethereum’, <https://coinmarketcap.com/currencies/ethereum/historical-data/?start=20190820&end=20200820>. Accessed: 20.08.2020.
- CoinMarketCap (2020c), ‘USD Coin’, <https://coinmarketcap.com/currencies/usd-coin/historical-data/?start=20190820&end=20200820>. Accessed: 20.08.2020.
- Cointelegraph (2020), ‘15% Correction Drops Bitcoin Price to USD 8,100 Days Before BTC Halving’. <https://cointelegraph.com/news/15-correction-drops-bitcoin-price-to-8k-two-days-before-btc-halving>  
Accessed: 20.06.2020.
- Compound Labs, Inc. (2020), ‘Compound’, <https://compound.finance/governance/comp>. Accessed: 01.08.2020.
- Cong, L. W., Li, X., Tang, K. and Yang, Y. (2019), ‘Crypto Wash Trading’, *Available at SSRN: <https://ssrn.com/abstract=3530220> or <http://dx.doi.org/10.2139/ssrn.3530220>* .
- DeFi Saver (2020), ‘Introducing 1-transaction CDP closing powered by flash loans’. <https://medium.com/defi-saver/>

introducing-1-transaction-cdp-closing-powered-by-flash-loans-8a83456226f4

Accessed: 20.06.2020.

Dune Analytics (2020), 'Dune Analytics', <https://www.duneanalytics.com>. Accessed: 16.07.2020.

dYdX (2020), 'dYdX documentation - solo protocol', <https://docs.dydx.exchange/#actions>. Accessed: 04.07.2020.

ERC20 token generator (2020), 'tokenmint', <https://tokenmint.io/app/#/token>. Accessed: 08.05.2020.

Ethereum Foundation (2020), 'py-evm Documentation Release 0.3.0-alpha.14', <https://readthedocs.org/projects/py-evm/downloads/pdf/latest>. Accessed: 08.05.2020.

Etherscan (B1) (2020*a*), 'Download Data (List of Verified Contract addresses with an OpenSource license)', <https://etherscan.io/exportData?type=open-source-contract-codes>. Accessed: 01.08.2020.

Etherscan (B1) (2020*b*), 'Ethereum Developer APIs', <https://etherscan.io/apis>. Accessed: 20.07.2020.

Etherscan (B1) (2020*c*), 'Etherscan', <https://etherscan.io>. Accessed: 16.07.2020.

flashmint (2020), 'flashmint.', <https://flashmint.app>. Accessed: 16.07.2020.

Frangella, E. (2020), 'Flash Loans, one month in', <https://medium.com/aave/flash-loans-one-month-in-73bde954a239>. Accessed: 14.08.2020.

fravoll (2019), 'Checks Effects Interactions', [https://fravoll.github.io/solidity-patterns/checks\\_effects\\_interactions.html](https://fravoll.github.io/solidity-patterns/checks_effects_interactions.html). Accessed: 21.07.2020.

Gudgeon, L., Perez, D., Harz, D., Gervais, A., Livshits, B. et al. (2020), 'The Decentralized Financial Crisis: Attacking DeFi'.

Hex (2018 -2020), 'Hex.win', <https://hex.win>. Accessed: 16.07.2020.

Hussey, M. and Tran, K. C. (2020), 'What is bZx?', <https://decrypt.co/resources/bzx-ethereum-defi-decentralized-finance-explained-guide>. Accessed: 20.07.2020.

investing.com (2020), 'Ethereum', <https://www.investing.com/crypto/ethereum/historical-data>. Accessed: 16.08.2020.

jay for curl on Github (2020), 'curl://', <https://github.com/curl/curl>. Accessed: 12.08.2020.

Julien Bouteloup for Flash Boys (2020), 'Flash Boys | Arbitrage DAO', <https://medium.com/@bneiluj/flash-boys-arbitrage-dao-c0b96d094f93>. Accessed: 08.05.2020.

Kevin Truong (2020), 'Collateral Swap Alpha', <https://collateralswap.com>. Accessed: 08.05.2020.

Khatri, Y. (2020), 'DeFi lending protocol bZx exploited, 'a portion of ETH lost' ', <https://www.theblockcrypto.com/linkedin/56134/defi-lending-protocol-bzx-exploited-a-portion-of-eth-lost>. Accessed: 14.08.2020.

Kyber Network (2019), 'Introduction', <https://developer.kyber.network/docs/Reserves-Intro/>. Accessed: 21.07.2020.

Leshner, R. and Hayes, G. (2019), 'Compound: TheMoneyMarketProtocol', <https://compound.finance/documents/Compound.Whitepaper.pdf>. Accessed: 21.07.2020.

Livnev, L. (2020), 'No Title', [https://lev.liv.nev.org.uk/pub/bzx\\_debug.txt](https://lev.liv.nev.org.uk/pub/bzx_debug.txt). Accessed: 16.07.2020.



- Marc Zeller for Aave Blog (2020), ‘Sneak peek at Flash Loans’, <https://medium.com/aave/sneak-peek-at-flash-loans-f2b28a394d62>. Accessed: 08.05.2020.
- Max Wolff (2018), ‘Introducing Marble - A Smart Contract Bank’, <https://medium.com/marbleorg/introducing-marble-a-smart-contract-bank-c9c438a12890>. Accessed: 08.05.2020.
- mikeshazer for Orfeed on Github (2020), ‘OrFeed’, <https://github.com/ProofSuite/OrFeed>. Accessed: 01.08.2020.
- money-legos (2020), ‘Flashloans On DyDx’, <https://money-legos.studydefi.com/#/dydx>. Accessed: 04.07.2020.
- mrdavey for Aave (2020), ‘EZ-Flashloan - A barebones solidity template for Aave’s flashloans’, <https://github.com/mrdavey/ez-flashloan>. Accessed: 08.05.2020.
- Nikolaev, I. (2019), ‘HEX HEX und die Kohle ist weg? – Was steckt hinter dem „besseren Bitcoin“?’. <https://cryptomonday.de/hex-hex-und-die-kohle-ist-weg-was-steckt-hinter-dem-besseren-bitcoin>. Accessed: 16.07.2020.
- PeckShield (2020), ‘bZx Hack Full Disclosure (With Detailed Profit Analysis)’. <https://medium.com/@peckshield/bzx-hack-full-disclosure-with-detailed-profit-analysis-e6b1fa9b18fc>. Accessed: 16.07.2020.
- PULSE, D. (2020), ‘DeFi Status Report Post-Black Thursday’, <https://defipulse.com/blog/defi-status-report-black-thursday>. Accessed: 14.08.2020.
- Qin, K., Zhou, L., Livshits, B., Gervais, A. et al. (2020), ‘Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit’, *arXiv e-prints* p. arXiv:2003.03810.

Synthetix (2020), ‘Litepaper’, <https://docs.synthetix.io/litepaper>. Accessed: 12.08.2020.

The GraphQL Foundation (2020), ‘A query language for your API’, <https://graphql.org>. Accessed: 20.06.2020.

TheGraph (2020), ‘APIs for a vibrant decentralized future’, <https://thegraph.com>. Accessed: 05.06.2020.

tokenmint (2019), ‘Mintable ERC20 TOKEN EXPLAINED’, <https://tokenmint.io/blog/mintable-erc20-token-explained.html>. Accessed: 08.05.2020.

Trustnodes (2020*a*), ‘Guy Flashloan Borrows \$30 Million ETH For Just \$100’, <https://www.trustnodes.com/2020/07/06/guy-flashloan-borrows-30-million-eth-for-just-100>. Accessed: 25.07.2020.

Trustnodes (2020*b*), ‘Hacker Makes \$360,000 ETH From a Flash Loan Single Transaction Involving Fulcrum, Compound, DyDx and Uniswap’. <https://www.trustnodes.com/2020/02/15/hacker-makes-360000-eth-from-a-flash-loan-single-transaction-involving-fulcrum-compound-dydx-and-uniswap>  
Accessed: 08.05.2020.

Uniswap (2020*a*), ‘Automated Liquidity Protocol.’, <https://uniswap.org>. Accessed: 16.07.2020.

Uniswap (2020*b*), ‘Flash Swaps’, <https://uniswap.org/docs/v2/core-concepts/flash-swaps>. Accessed: 16.07.2020.

Uniswap (2020*c*), ‘How Uniswap works’, <https://uniswap.org/docs/v2/protocol-overview/how-uniswap-works>. Accessed: 16.07.2020.

Uniswap (2020*d*), ‘Smart Contract Integration - Flash Swaps’, <https://uniswap.org/docs/v2/smart-contract-integration/using-flash-swaps>. Accessed: 16.07.2020.

Uniswap (2020*e*), ‘Smart Contract Integration - Providing Liquidity’, <https://uniswap.org/docs/v2/smart-contract-integration/providing-liquidity>. Accessed: 16.07.2020.

Uniswap (2020*f*), ‘Uniswap V2 Overview’, <https://uniswap.org/blog/uniswap-v2>. Accessed: 16.07.2020.

Uniswap (2020*g*), ‘Uniswap Whitepaper’, [https://hackmd.io/C-DvwDSfSxuh-Gd4WKE\\_ig](https://hackmd.io/C-DvwDSfSxuh-Gd4WKE_ig). Accessed: 16.07.2020.

Waskom, M. (2012-2020), ‘seaborn: statistical data visualization’, <https://seaborn.pydata.org>. Accessed: 16.08.2020.