Towards Understanding Flash Loan and Its Applications in DeFi Ecosystem

Dabao Wang∗
Zhejiang University
Hangzhou, Zhejiang, China
dabao.wang@monash.edu

Siwei Wu
Zhejiang University
Hangzhou, Zhejiang, China
wusw1020@zju.edu.cn

Ziling Lin
Zhejiang University
Hangzhou, Zhejiang, China
linziling@zju.edu.cn

Lei Wu†
Zhejiang University
Hangzhou, Zhejiang, China
lei_wu@zju.edu.cn

Xingliang Yuan
Monash University
Melbourne, Victoria, Australia
xingliang.yuan@monash.edu

Yajin Zhou
Zhejiang University
Hangzhou, Zhejiang, China
yajin_zhou@zju.edu.cn

Haoyu Wang
Beijing University of Posts and Telecommunications
Beijing, China
haoyuwang@bupt.edu.cn

Kui Ren
Zhejiang University
Hangzhou, Zhejiang, China
kuiren@zju.edu.cn

ABSTRACT
Flash Loan, as an emerging service in the decentralized finance ecosystem, allows traders to request a non-collateral loan as long as the debt is repaid within the transaction. While providing convenience, it brings considerable challenges that Flash Loan allows speculative traders to leverage vulnerability of deployed protocols with vast capital and few risks and responsibilities. Most recently, attackers have gained over $15M profits from Eminence Finance [53] via exploiting Flash Loans to repeatedly swap tokens (i.e., EMN and DAI).

To be aware of foxy actions, we should understand what is the behavior running with the Flash Loan by traders. In this work, we propose ThunderStorm, a 3-phase transaction-based analysis framework, to systematically study Flash Loan on the Ethereum. Specifically, ThunderStorm first identifies Flash Loan transactions by applying observed transaction patterns, and then understands the semantics of the transactions based on primitive behaviors, and finally recovers the intentions of transactions according to advanced behaviors. To perform the evaluation, we apply ThunderStorm to existing transactions and investigate 11 well-known platforms. As the result, 22, 244 transactions are determined to launch Flash Loan(s), and those Flash Loan transactions are further classified into 7 categories. Lastly, the measurement of financial behaviors based on Flash Loans is present to help further understand and explore the speculative usage of Flash Loan. The evaluation results demonstrate the capability of the proposed system.

KEYWORDS
Blockchain, Flash Loan, DeFi, Measurement

1 INTRODUCTION
Decentralized finance, aka DeFi, has drawn much attention in recent years. Up to 20th Oct 2020, the total value locked of DeFi has reached 11 billion USD [18]. The popularity of DeFi is partially due to Flash Loan (i.e., non-collateral loan), which is impractical in the traditional centralized finance system.

However, the introduction of Flash Loan is a double-edged sword. On the one hand, it does bring in convenience [45] and facilitate the prosperity of DeFi. Traders without much capital can launch arbitrage, liquidation and asset swapping with Flash Loan. For instance, when traders discover a considerable price difference among tokens between decentralized exchanges (DEXes), Flash Loan can “generously” lend traders a considerable amount of capital to maximize profits and require them to repay at the end.

On the other hand, Flash Loan may raise risks [22] [21] to DeFi. With a Flash Loan, users can launch actions such as exchange, lending and borrowing, with a vast amount of assets that they do not have. In early 2020, two infamous incidents [22] [21] also cause a huge loss to bZx [9] (A lending & borrowing platform which also provides margin trading.). They take the advantage of the Flash Loan and manipulate the market price to make considerable profits of 829.5 thousand USDs and 1.1 million USDs, respectively [44]. Most recently, a trader borrowed 15M DAI [39] in a Flash Loan to gain over $15M through repeatedly swapping tokens on the EMN [52] pool.

As such, there is an urgent need to demystify the Flash Loan ecosystem, and understand the impact of potential security threats. Otherwise, it is impossible to provide effective mitigations without a comprehensive understanding. Unfortunately, few studies have been proposed to serve this purpose. Specifically, previous studies mainly focused on profit optimization [16, 44] and oracles [37] used in DEXes. To the best of our knowledge, none of them systematically demystified Flash Loan and its applications in DeFi ecosystem.

1 Aka TVL, a standard to measure the total worth of the collateral deposited by users.
Namely, we still lack a comprehensive understanding of Flash Loan, especially the potential security impacts to the DeFi ecosystem. 

**Our approach.** In this paper, we take the first step to systematically study Flash Loan in a comprehensive way. To this end, we propose a transaction-based analysis framework named ThunderStorm to demystify Flash Loan by measuring all existing transactions. ThunderStorm is composed of three components, including Flash Loan Identifier, Primitive Classifier and Advanced Classifier, which provide a 3-phase filtering to produce three classes of transactions step by step. These classes of transactions will facilitate revealing the characteristic of Flash Loan.

Specifically, to support the analysis, we first collect and investigate 11 representative platforms to extract function/event signatures to feed into ThunderStorm. After that, with the help of collected patterns, ThunderStorm is capable of analyzing the existing transactions. Specifically, in the first phase, Flash Loan Identifier locates all Flash Loan transactions until 10th Oct 2020 (Section 5.2). Then, in the second phase, Primitive Classifier groups these transactions into four primitives based on the behaviors acted within Flash Loan. Next, in the third phase, Advanced Classifier further categorizes four groups of transactions gained in the second phase into three advanced behaviors to reflect the intention of using the Flash Loan (Section 5.3). Last, ThunderStorm measures each class of transactions and presents their distribution and intersection to understand the frequency of each behavior utilized by users. We further articulate the challenges to deeply analyze advanced behaviors of Flash Loan transactions (Section 6).

**Results.** We have applied the proposed approach to analyze 4 representative Flash Loan providers, including Aave [4], bZx [9], dYdX [19] and UniswapV2 [47]. For transactions of Flash Loan identified in the first phase, 22,244 transactions were filtered from total 863,504, 142 transactions mined until 10th Oct 2020. Among them, we discovered that over 60% of Flash Loan transactions were triggered by only 3.5% of borrowers. Lately, we conducted a behavior-based analysis on all filtered Flash Loan transactions and classified them into total 7 groups with primitives and advanced behaviors. Totally, 2,331 transactions were collected. Furthermore, we concluded 3 challenges, i.e., 1) **How to collect historical information;** and 2) **How to interpret extracted information with meaningless parameter names;** and 3) **How to summarize comprehensive arbitrage behaviors to understand the behavior launched by users.**

**Contributions.** This paper provides the following main contributions:

- To the best of our knowledge, this paper is the first systematic study to understand Flash Loan. We propose a three-phase analysis framework to serve the need.
- We conduct full-chain measurements on financial behaviors launched with the Flash Loan in DeFi ecosystem. As far as we know, this is the first work to give a measurement for users’ movements behind the Flash Loan based on real-world data.
- We analyze the distribution of the aforementioned behaviors, and exhibit the challenges that hinder us to further interpret the behavior with extracted information.

The rest of the paper is arranged as follows. Section 2 introduces the background of some concepts on Ethereum and primitive behaviors on DeFi. Section 4 presents the workflow of ThunderStorm in details. Section 5 discusses the distribution of filtered results and how ThunderStorm help us to understand speculative usage of Flash Loan in the real world. Section 6 shows the identified challenges that hinder users from understanding the behavior behind Flash Loan. The related work is exhibited in Section 7. Finally, we conclude the paper in Section 8.

## 2 BACKGROUND

The introduction of the blockchain technique [42] has changed the financial ecology in the world. Especially with the invention of Ethereum [48], there has been a wave of developing the decentralized application (DApp). Smart contracts, as the basis of DApps, enable a transparent environment and become an essential component for the development of DeFi.

### 2.1 Common concepts on Ethereum

In order to make this work easy to understand, We first introduce a few common concepts on Ethereum.

**Account:** Ethereum is an account-centric blockchain system. There are two types of accounts: External Owned Account (EOA) and smart contract account (smart contract in short). The main difference between them is that EOAs are controlled by their private keys and smart contracts are controlled by their codes. Basically, an EOA is created with the generation of the public and private key pair, and a smart contract is always created by an EOA or another smart contract. Both EOAs and smart contracts are identified by their addresses, like 0x16431837a35b54697562ba5d9b7575d25b721c3.

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

**Transaction:** All actions on Ethereum are based on transactions. A transaction may have three purposes: transferring Ether, invoking a function of a smart contract and deploying a smart contract. Transactions are normally initiated from EOAs, but a transaction invoking a smart contract may trigger more transactions that are initiated from the smart contract, which depends on the smart contract’s code. In order to distinguish them, we always refer the former as external transaction and the latter as internal transaction. The word “transaction” written individually in the remaining of the paper indicates the collection of an external transaction and internal transactions triggered by it.

**Function/event:** The function of smart contract is identified by the function signature, which is the first four bytes of the hash value (SHA3) of the function name with parenthesized list of parameter types. If a user sets a function signature in front of the call data of a transaction, then the corresponding function of the callee smart contract will be invoked. Smart contracts’ developers usually leverage event to record critical information. For example,
ERC20 token standard specifies an event *Transfer* to record the spender, receiver and amount for a ERC20 token transfer. Similarly, an event is identified by the hash value (SHA3) of the event name with parenthesized list of parameter types. When an event is triggered, a log with event hash is recorded in Ethereum. In table 1, we take advantage of functions’ signature and events’ hash to identify various behaviors.

2.2 Finance Behaviors in DeFi

In Ethereum, DeFi is formed with a group of open-source protocols deployed as smart contracts. Compared to the traditional centralized finance system, decentralized finance is a transparent and permissionless finance ecosystem without relying on intermediaries such as banks.

**Flash Loan.** To request a loan in DeFi platforms, the borrower is normally required to deposit self-owned collaterals (i.e., digital cash or tokens). However, with the state reverting feature of Ethereum, a new service called *flash loan* is developed to enable a non-collateral borrowing service [44]. The flash loan can provide users a considerable amount, which is limited by the total locked value in the pool, of asset, as long as borrowed assets could be repaid within internal transactions triggered by the flash loan service. To our best knowledge, there are four platforms [4] [9] [19] [47] supporting flash loans with or without a certain percentage of the trading fee.

With the development of the flash loan service, two hacks [21] [22] on 15th and 18th Feb 2020 caused a serious capital loss via leveraging the flash loan service provided by dYdX [19] and bZx [9], respectively. The attacker only spent a small amount of transaction fees to gain a considerable amount of profits.

**Decentralized Exchange (DEX).** To exchange two types of crypto assets in the decentralized circumstance, two exchange modes (LOB & AMM) are regularly leveraged by known DEX platforms. *Limit order book* (LOB) DEXes such as MakerDao [40], match a pair of buy and sell orders to perform the exchange for different types of crypto-assets with a compatible price agreed by the buyer and seller. As one of the most impactful innovations in DeFi, Automated Market Maker (AMM) DEXes like Uniswap [47] and Balancer [6] consist of various liquidity pools supported by liquidity providers who are normally required to deposit two different tokens for liquidity equilibrium. Different from LOB, AMM DEXes allow traders to exchange their crypto-assets with high flexibility. However, a considerable slippage could happen in Constant Function Market Maker (CFMM) DEXes, which are the most common class of AMMs, if there exists a considerable amount of exchange in a small liquidity pool.

**Lending & Borrowing.** As the replacement of the bank in DeFi, lending platforms share interests for depositors locking their capital in the liquidity pool and also provide a collateral loan for borrowers. To keep the balance of the loss, liquidation is applied once the cost of the borrowed out money is slipping.

**Margin Trade.** Margin trade is one of the most common financial derivatives. It is a risky behavior for traders to launch profit making. Moreover, the buyer normally plays it with a leverage to amplify the capital for more benefits, however, it speeds up the loss of their collateral as well. On 15 Feb 2020, a trader launches a margin trade with the vast capital borrowed from flash loan. This action manipulates the market price and creates a considerable profit-making opportunity.

**Liquidation.** The lending & borrowing, and margin trade platforms normally require traders to deposit more collateral than the borrowed assets with a certain ratio. To prevent the potential loss caused by price drop of traders’ deposited collaterals, liquidation will be triggered. Platforms allow liquidators to launch auctions on trader’s collateral if the boundary collateralization ratio(collateral value/debt value) is reached. As the consequence, traders could have a considerable loss (13% punishment applied in MakerDAO platform for CDP) on their collateral. In contrast, the platform provides a discount and encourages liquidators to buy all liquidable collaterals. Liquidation could happen everyday due to the price fluctuation in cryptocurrency market. On 12th March 2020, due to the large-scale turmoil in global financial market, the price of bitcoin is affected and dropping to the half or its original price as well as ether plunges dramatically too. This considerable drop causes traders’ collateralization ratio falls below its minimum boundary as well as vast liquidations are triggered. Overall, exceeding 23M USD debt is liquidated on this single day. 3994 liquidations were triggered in MakerDAO and over 5M and 550K USD worth of crypto assets were liquidated on Compound and Aave, respectively.

3 A MOTIVATION EXAMPLE

Here, we use a real accident as the motivating example to lead the application of Flash Loan and demonstrate the difficulty and complexity to understand the whole picture. On February 15, 2020, an accident involving dYdX [19], bZx [9], Compound [14] and Uniswap [47] made roughly $335,880 [2], which known as bZx Hack. Ethereum analyst usually use Bloxy [7] to help analyze transactions. However, the transaction [21] in this accident contains a lot of internal transactions, which makes deeply understanding this accident difficult. As shown in Figure 1, the invocation graph provided by Bloxy is too complicated to identify behaviors in this accident.

2Kyber is a DEX aggregator, and it chose Uniswap as DEX in this attack. Therefore, we removed Kyber for simplicity.
We get the following two observations from the above accident. First, the huge initial capital is an essential prerequisite for this accident, and Flash Loan service can help users meet this prerequisite. Therefore, clarifying all applications of Flash Loan is an intuitive idea. Second, a DeFi platform may contain dozens of smart contracts with lots of interactions (with other DeFi platforms), which make DeFi-related transactions too complicated to understand. As Figure 1 shows, transaction involving DeFi platform is difficult to analyze, which motivates us to propose a better approach to help to identify and analyze these DeFi-related (or Flash Loan-related) behaviors.

4 METHODOLOGY

4.1 Overview

The purpose of our work is to understand the Flash Loan and its applications in DeFi ecosystem. Without this knowledge, it is hard to understand the attacks that have occurred and propose the corresponding defenses. However, due to the large number of transactions in Ethereum and multiple smart contracts involved, our work must solve the following three challenges.

- First, we need to filter the transactions that are related Flash Loan efficiently, from billions of transactions in Ethereum and hundreds of smart contracts involved.
- Second, we need to quickly understand the semantics of the transactions and their primitives, e.g., performing exchanges, and margin trading.
- Third, we need to further understand the high-level intentions of the advanced behaviors involved in the Flash Loan related transactions, e.g., arbitrage and swapping.

Accordingly, we propose a three-phase solution to address the previous challenges.

- First, to filter transactions, we first define the transaction patterns used in the Flash Loan. The pattern includes transaction events, single function invocation or the chain of multiple function invocations (or transaction events) between smart contracts. Note that, we need to define the patterns for each platform that provides the Flash Loan service.
- Second, we leverage the transaction pattern to understand the semantics of the transaction. This provides us to construct four primitives used in DeFi applications, including exchange, lending and borrowing, margin trading and liquidation. For instance, if the borrow function of the smart contract of the AAVE is invoked, then we know that the lending and borrowing behavior has occurred.
- Third, we understand the intentions of the transactions by the combination of the recovered primitives.

We propose a three-phase framework named ThunderStorm to perform the analysis. Figure 2 depicts the workflow of ThunderStorm, which consists of 3 major components, i.e., FlashLoan Identifier, Primitive Classifier and Advanced Classifier. Each aims to solve one of the previous challenge. We will elaborate on them in the following sections.

4.2 Flash-Loan Identifier

Our approach uses transaction patterns to identify Flash Loan related transactions and classify the primitives. In our work, a transaction pattern could consist of the transaction event (that a specific event has occurred), the invocation of a function (through an internal transaction) and the chain of the invocation of functions between smart contracts.

However, unlike the ERC20 token [1], there is no standard for a DeFi application to use the standardized function names to implement the pre-defined functionality. Hence, we manually studied the popular DeFi platforms on defiprime [17], and constructed the transaction patterns.

Table 1 summarizes the platforms, and corresponding events and functions that are used to identify their provided services. For example, Aave implements a public function named flashloan to allow users to utilize its Flash Loan service. Thus when flashloan is successfully called, the event FlashLoan will be triggered to record related information. We locate behaviors using the Flash Loan of Aave via the function signature of flashLoan and event hash of FlashLoan.

In the following, we will first elaborate on existing Flash Loan providers and the patterns to filter related transactions.

4.2.1 Determine Flash Loan Providers. Through investigating some sources, including the famous web source [17, 18] and open source code, platforms, i.e., Aave, bZx, dYdX and UniswapV2 are determined as Flash Loan providers. In this section, we give an introduction of each platform and describe how they provide Flash Loan with fees.

- **Aave** As the second large lending & borrowing platform in DeFi, up to date, Aave has over $1.12B in total value locked before Oct 2020 [18]. As the first platform officially providing the Flash Loan for users, Aave offers 17 different types of crypto-assets including DAI, BAT, ETH and etc. To use Flash Loan in Aave, the platform charges 0.25% of the borrowed asset as a fee from users.

Only UniswapV2 provides Flash Loan service in Uniswap [47]
Table 1: Function signature or event hash of platforms

| Platform     | Function Name | Signature | Event Name      | Event Hash                                                      |
|--------------|---------------|-----------|-----------------|----------------------------------------------------------------|
| AAVE         | flashLoan     | 0x5cffe9de| FlashLoan       | 0x5b8f46461c1dd69fb968f1a003acee221ea3e19540a350233b612ddb4343b55 |
| bZx          | flashBorrowToken | 0x66fa576f| Pair Created    | 0x03d648bf0b6fb8a0134a3b9275ac5859d315f0dd8355cdedfe31af2a8b09e9 |
| Uniswap V2   | swap          | 0x022c0d4f| Swap            | 0x78ad95f4a09c46651d0da58f257f6e13ce7657fbd8de3e130840159d822 |
| dYdX         | -             | 0x86fa576f| LogOperate      | 0x19b10b67ab395d16213814868010057f9353ee7f1ee433e3c3f83f0b0   |
|             |               |           | LogWithdraw     | 0x8c380fcb269f1726909c834bffdf1aef169624a148687d766e54a2f18c7d64 |
|             |               |           | LogCall         | 0x3ab38c4d838ee26652b277d3666d6b7c6c6a4d067cc54cf56ac38de0b5c30098 |
|             |               |           | LogDeposit      | 0x2badb8c9508baf2c27b30f2a7e8a0888f8665e0945cd30510b0e72019f8f |
|             |               |           |                 |                                                                |

| Platform     | Function Name | Signature | Event Name      | Event Hash                                                      |
|--------------|---------------|-----------|-----------------|----------------------------------------------------------------|
| Uniswap V1   | TokenPurchase | 0x66fa576f| TokenExchange   | 0x19b10b67ab395d16213814868010057f9353ee7f1ee433e3c3f83f0b0   |
|              | ETHPurchase   | 0x7360f191c3e3b93a62307641d176b70292477a3469545b359894237050 |
|              | LOG_SWAP      | 0x90f8f6ee5f6c6c9c3b360973819f324a0f10881335434289700e10d3378 |
| LineCh       | swap          | 0x08f390a79| Swapped         | 0x2ce4e36883059280b673943853afed9b302615d0b77428a4f628a8556 |
| Synthetx     | Exchange      | 0x8b3e96f2b889fa77c53c91b04a0f5563f63f7196477075225add79140 |
|              |               |           | TokenExchange   | 0x184999db6a301bca28834837f4ed1963752d5f1727ae7c87b76165682a39 |
| Kyber        | executeTrade  | 0x08f390a79| KyberTrade      | 0x3d30ca399b43507e6ce56a2935c45eb89c9a50c2796960dc08c4a38736e6c |

| Platform     | Function Name | Signature | Event Name      | Event Hash                                                      |
|--------------|---------------|-----------|-----------------|----------------------------------------------------------------|
| AAVE         | Borrow        | 0xe774f6728e5585a16e100c6ab9c3cbb680593093f5d43356489801453923953f9 |
|              | Repay         | 0x871b0f140b73d93a3d5b515e3a5d2a221b042a2e387e5a85851e1b5cf0657737 |
| dYdX         | Redeem        | 0x12c57b1c73a2c2a2e46613e478a6bb3dd1a46857a7b5292432b59710c820 |
|              | RedeemUnderlying | 0x96e5599e3e96637623040d771e7b7929b4c499c4d41f7b765c3c6e6c7 |
|              | Burn          | 0x86e15dd787cd7f842b7788bcb5b9b9395e86030e046e5a2cef7c72c70f1e657e |
|              | Deposit       | 0x85dfc033a3e2eb9915b1577d9c1f9b8556d6b33ff5b7928368d2e8a5373a |
|              | Mint          | 0xb4c03061fb5b7f6ed73689df59a582f200df0a09f6e70d3333badd625828d17oac8c |
|              | LiquidationCall| 0x7430337678f4378f4df6d7225ce2b09d77ef869b1a902a585f6d40e297b644 |
| Compound     | RepayBorrow   | 0x13e63d8864d4e1ed6a46f845c6d7e54128837d575ea92a2ac6ccca8c894ab80 |
|              | Mint          | 0x1e2a22cb0344c4a61584a5d6c6de668a591fe25e5affea7db355478665f3c6a21 |
| Maker DAO    | anonymous     | 0x1760887030| 0x6529af30e1a15b48498c0830b7982368d2e8a5373a |

| Platform     | Function Name | Signature | Event Name      | Event Hash                                                      |
|--------------|---------------|-----------|-----------------|----------------------------------------------------------------|
| AAVE         | LiquidationCall | 0x866a6757f5d85b1f29f38f53a981cd8e5a12ce41b902cf73f506ee3696b237 |
| Compound     | LiquidateBorrow| 0x1968933172176a2c2d257008b8d89c7c819c485b2a1565390df650326f2 |
| dYdX         | LogLiquidate  | 0x1b9e65b3599871d74b1af1fcb810b13b653b90f7c4631b901eb762da6e76d7c7 |
| opyn         | -             | 0xcab8e1abb9f8235c6db989c1f853336dc941ee47f79b8c1be83687ee549e66a |

**dYdX**

- **dYdX** is a lending and borrowing platform with over $24M total value locked up to date. Apart from lending and borrowing service, it provides the margin trading service as well. As for the Flash Loan service, there is no official publication for Flash Loan in *dYdX*. However, *dYdX* allows users to perform Flash Loan logic, which is to call the function 'withdraw' before 'deposit', by starting with provided 'operator' function. In addition, there is no extract fee charged except the basic transaction gas.

- **bZx** Similar to *dYdX*, *bZx* provides both lending and borrowing and margin trading, but with less total value locked (around $5M).
eral identifying process in function Identifying pattern p; after we precisely present the filtering logic to identify Flash Loan transactions, we use the patterns used by each Flash Loan provider. More-over, we precisely present the filtering logic to identify Flash Loan transactions. For each pair of (p, t), once we can identify the pattern p in the transaction t, function ExtractInfo(t,p) which is revealed in the next subsection, will be triggered to collect information of Flash Loan. Furthermore, the identified transaction t and corresponding information info will be appended to the set fT.

For each Flash Loan provider, the logic applied in the condition of if loop is different. We will elaborate for each provider in the following.

**Aave**: Specifically, Aave exposes an external function called flashLoan in the smart contract AAVELendingPool for users to utilize the Flash Loan service. Moreover, once the Flash Loan is executed successfully, a corresponding event will be emitted so that we can further confirm the existence of Flash Loan in a transaction. To identify Flash Loan of Aave in a transaction, we firstly search the hash value of the emitted event among all events of current transaction. Then we check the "from" address with the contract address $^5$ of AAVELendingPool: If both conditions, i.e., event is found and the "from" address is confirmed, this transaction definitely contains Flash Loan from Aave.

**bZx**: Similar with Aave, locating the transactions including Flash Loan requires an identification of the invocation of borrowing any type of iToken from bZx. With the function signature of flashBorrowToken and all iToken’s addresses, once the function flashBorrowToken is invoked, then this transaction must include Flash Loan supported by bZx.

**UniswapV2**: As aforementioned, Flash Loan is inherited as the flash swap feature in UniswapV2. However, based on the protocol of UniswapV2, a normal crypto-assets swap and a flash swap are both operated via invoking a same function swap. To differentiate them, we check the existence of the payback action, which is simply the transfer or transferFrom function sending the token back to the contract operating swap function.

First, since the swap function is only invoked in contract UniswapV2Pair created by UniswapV2Factory, we collected all pair contracts via identifying the address of UniswapV2Factory associated with the hash value of the event PairCreated. As the result, we get a group of filtered addresses for the next round of filtering. Second, with the help of the function signature of swap, the transaction triggered by this function could be located easily. As shown in Figure 3, the function uniswapV2Call is invoked in swap to launch the payback action. Moreover, to confirm the existence of the payback action, three conditions should be fulfilled, including 1) existing call-data in the function uniswapV2Call; 2) all internal transactions triggered by the calling data must present a transfer transferFrom function; 3) in the transfer or transferFrom function, we get a group of filtered addresses for the next round of filtering.

$^250\text{K})$. As an ERC20 token based protocol, bZx launches different services with different tokens: BZXToken, iToken, and pToken $^9$. As for its Flash Loan service, bZx does not declare the Flash Loan officially and it requires no fees except basic transaction gas.

- **UniswapV2**: As one of the most well-known DEX, Uniswap has the highest total value locked (over $2.7\text{B}$) among all DEXes. Recently, the latest protocol UniswapV2 is published by the community and a new feature called flash swap is introduced. This feature inherits the property of Flash Loan which allows users to get crypto-assets in advance and repay them after a series of actions. In terms of the fee, compared to the aforementioned Flash Loan providers, it charges the highest percentage(0.3%) of fees based on users’ borrowed asset.

### 4.2.2 Identify Flash Loan Transactions

To effectively identify Flash Loan, we use the patterns used by each Flash Loan provider. Moreover, we precisely present the filtering logic to identify Flash Loan among aggregated transactions. Furthermore, we structure the general identifying process in function **Identifying** presented in Algorithm 1.

**Algorithm 1: The Algorithm to Identify Flash Loan Transactions**

```
ExtractInfo(t, p)
inputs: Flash Loan transaction t; Identifying pattern p;
output: Expecting information info
info ← ∅;
foreach para ∈ p.Parameters do
  if Identified by Event from p then
    info[para] ← Event[para] in t;
  else
    info[para] ← Function[para] in t;
foreach Internal transaction it ∈ t do
  if it executes flash_loan_borrow then
    info[IntStart] ← it.index;
  else if it executes flash_loan_repay then
    info[IntEnd] ← it.index;
    break;
return info;

Identifying (P, T)
inputs: Collected patterns P for Flash Loan; Collected existing transactions T;
output: Flash loan transactions fT;
fT ← ∅;
foreach t ∈ T do
  foreach p ∈ P do
    info ← ∅;
    if Pattern p is identified in t then
      info ← ExtractInfo(t, p);
      fT[t] ← info;
      break;
return fT;
```

Through running the function Identifying, it returns a set of Flash Loan transactions fT with input of pre-collected patterns P and existing transactions (up to 10th Oct 2020) T. Specifically, we firstly initiate an empty set fT, then we iterate each pattern p of P for each transaction t of T. For each pair of (p, t), once we can identify the pattern p in the transaction t, function ExtractInfo(t,p) which is revealed in the next subsection, will be triggered to collect information of Flash Loan. Furthermore, the identified transaction t and corresponding information info will be appended to the set fT.

4. flashLoan(address _receiver, address _reserve, uint256 _amount, bytes calldata _params)
5. 0x0398ec7346dc622edc5ae82352f02be94e62df119
6. They are interest accumulating tokens that continuously go up in value as you hold them
7. flashBorrowToken(uint256 borrowAmount,address borrower,address target,string signature,bytes data)
8. BZXToken(uint256 borrowAmount,address borrower,address target,string signature,bytes data)
the receiver must be the address of current pair contract which triggered the swap function.

Once these three conditions are fulfilled, the flash swap (i.e., Flash Loan in UniswapV2) can be confirmed in the transaction.

dYdX: Flash Loan must be completed in a single transaction in dYdX. As aforementioned, Flash Loan in dYdX is constructed by starting calling the operate function. Normally, a transaction is created by a function call. In dYdX, it allows the combination of functions to create one meta transaction. Particularly, a meta transaction is constructed with functions: Operate, Withdraw, callFunction, and Deposit in order. In Figure 3, the basic structure of a meta transaction for Flash Loan is present and we use function Call to replace the function callFunction.

Furthermore, for each function mentioned above, they all have a corresponding event and these events, which are LogOperate, LogWithdraw, LogCall, and LogDeposit, are defined as patterns and specified in Table 1. Therefore, as long as we can identify ordered events based on either of the following two invocation flows, then the transaction certainly has Flash Loan.

- LogOperate → LogCall → LogWithdraw
- LogOperate → LogWithdraw → LogDeposit

4.2.3 Extract the Position information in Internal Transactions (Information Extracting). To better understand associations with the Flash Loan, extra information is also needed.

In Algorithm 1, once the Flash Loan is confirmed in function Identifying, the function ExtractInfo(p, t) is triggered instantly to extract required information listed in Table 2 for Flash Loan. For details, ExtractInfo(p, t) returns the expected information. More specifically, first of all, all parameters mentioned in functions used to identify Flash Loan will be aggregated. Apart from the parameter, the index of the start and the end of internal transactions covered in the Flash Loan are collected. These two indexes are used to limit our further analysis (Section 4.3) within the transactions that are related with Flash Loan.

```
callFunction(address sender,Account.Info memory account,bytes memory data)public
```

4.3 Primitive Classifier

After filtering Flash Loan related transactions, our next step is to recover the semantics of the transactions to four primitives. A primitive here is used to describe the most straightforward services provided in DeFi, including decentralized exchange (DEX), lending and borrowing, margin trading and liquidation. These primitives are the foundations for the next step to classify and understand the advanced behaviors.

Note that, in this step, we use the identified Flash Loan transactions in the previous step as the input, since our purpose is to understand Flash Loan related behaviors. The basic idea is still using the patterns defined for each primitive to filter the transactions. The method is similar with the previous step, but with different patterns used. Table 1 shows the patterns for each primitive.

4.4 Advanced Behavior Classifier

Advanced Classifier aims to reveal the high-level action from traders’ initial purposes and intention like arbitrage, preventing positions from liquidation and swapping. This phase takes transactions returned by Primitive Classifier as the input. We take different strategies to identify them.

4.4.1 Arbitrage. Generally speaking, the opportunity of arbitrages are due to price and interest rate fluctuation of assets. The DeFi normally reacts slower for events happening in the network than the real world market. Therefore, traders can take the advantage of the inefficiencies of the market to buy/deposit and sell/withdraw the asset across different platforms.

We analyze the arbitrage strategies with two key components, i.e., price and interest rate. For price arbitrage, since the price synchronization latency in DeFi, it allows traders to make gains via trading the same asset with price difference among different platforms. On the other hand, interest rate arbitrage normally happens in lending and borrowing platforms. Depositing capital in platforms providing higher interest rate could give users more benefits.

Here we mainly analyze the most common arbitrage, in which traders "smartly" leverage the price difference. Moreover, speculative traders might even manipulate the price leveraging the constant-value mechanism deployed by AMM DEXes [47], which...
maintains a constant proportion of value in each asset of the portfolio. Arbitrage behavior transactions are identified based on the DEX transactions filtered at the previous phase.

As shown in Figure 4, we present two cases to describe the workflow of arbitrage identification. On the left, a trader swaps out DAI by providing ETH within Kyber [34]. Later on, the same trader launches another swap to replace DAI in 1.inch [3] again. Therefore, in our strategy, we define the pattern of arbitrage by searching at least two trades launching with a same trader.

4.4.2 Anti-Liquidation. Liquidation happens nearly every day in DeFi. Through experiencing the black Thursday in early 2020, protecting existed collateral could not be underestimated by traders because of the considerable punishment. We collect three ways [45] which can prevent liquidations from happening. Furthermore, for the ease of understanding, we rename this category of actions as Anti-Liqudation. When a premonition of the price drop is encountered, traders can have three options listed below:

1. Repaying all debts with the required fee to save the collateral. This option saves the trader from devaluation at once, but the loss is fixed as well.
2. Depositing more assets into the platform to increase the collateralization ratio. This method gives the trader a chance to manage their loss or even revive. However, it might lead to a more serious loss if the price drop is continuing.
3. Instead of completely unwinding the position, traders can partially payback their debt with their deposited collateral. This partially payback does not only protect users’ collateral by increasing the collateralization ratio, traders can have collateral remaining in the pool.

In this work, we only focus on the anti-liquidation behavior of a well-known third-party platform, i.e., DeFi Saver [45]. DeFi Saver, which is known as CDP Saver, is originally an advance management platform dashboard for CDP in MakerDAO [40], as well as capital and portfolio in Compound [15]. It provides automation solutions including decreasing loan ratio to prevent users from liquidation. On 1st Oct 2020, DeFi Saver announced the adoption of Aave Flash Loan service is finally done, which means that DeFi Saver now can launch an instant saving for users’ capital due to the price fluctuation. Moreover, compared to the punishment of liquidation, DeFi Saver only charge a few amounts of fee for users. To locate such anti-liquidation behavior launched by DeFi Saver, we can easily leverage its corresponding event identify in transactions.

4.4.3 Swapping. Through monitoring the price and timely swapping capitals might help users gain significant profit in DeFi because of its unpredictable and changeable market attributes. A low charged interest rate, a low bounded collateralization ratio or a better reward to the liquidity supplier can attract traders to launch such an action to refinance their capitals to gain the most profits. We combine three types of actions, i.e., collateral swapping, loan swapping and platform swapping, as swapping in our work. Among them, the collateral swapping changes the collateral types but borrows the same coin, while the loan swapping keeps the collateral with different assets borrowed out. As for the platform swapping, the trader closes a loan in one platform, draws out the collateral and opens a new vault in another platform. These actions require the traders to pay back the debts to go further. However, the loan borrowed by traders is normally used instantly. Instead of taking time to organize capital to swap, with the help of Flash Loan, traders can directly launch swapping.

All classified lending and borrowing transactions are used as the input for swapping identification. Moreover, in this work, we performed collateral swapping identification associated with the patterns collected in MakerDAO as well as loan swapping identification with the required pattern in Compound and Aave.

- **Collateral Swapping.** Achieving collateral swapping requires two compulsory actions: redeeming old collateral and opening a new loan. The basic idea of the behavior is to close the old loan and open a new loan. The asset borrowed from the flash loan can be either used to repay the debt to redeem the original collateral or to open a new loan. More specifically, the patterns required in the collateral swapping identification process with MakerDAO expresses two main actions: redeeming original collateral and depositing the new type of collateral. Once two actions are identified with the required pattern, two collateral asset types will be extracted. Then the collateral swapping will be finally confirmed in the flash loan transaction.

- **Loan Swapping.** Different from collateral swapping, there is no change needed for collateral to achieve such behaviors. The borrowed assets from the Flash Loan are used to repay the loan. Furthermore, with the original collateral, another type of loan will be opened. The classifying process is quite similar to the collateral swapping apart from the actions used to search. Specifically, two actions, i.e., repaying the loan back and borrowing a new loan, are constructed. On the other hand, the assets which pay back the old loan and the asset being borrowed should also be identical to further confirm the loan swapping behavior.

- **Platform Swapping.** Similar to the collateral swapping, an old loan must be close and a new vault will be opened. But these two actions happen in two platforms. Therefore the patterns of paying back the debt and drawing out the collateral in a platform, as well as depositing in another platform are checked to confirm such a behavior.

| Parameter Types | Flash Loan | Primitive Behaviors |
|-----------------|------------|---------------------|
| **Service Provider** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Runner** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Receiver** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Asset In** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Asset Out** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Amount In** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Block Number** | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Transaction Index** | ✓ | ✓ | ✓ | ✓ | ✓ |

*The Service Provider includes the provider of Flash Loan and other services. The Runner indicates a trader or a liquidator.*
5 EVALUATION

In this section, we will provide an in-depth evaluation based on our proposed approach discussed in Section 4. Specifically, to facilitate the understanding of Flash Loan, we seek to answer the following research questions:

RQ1 How Flash Loan works with different platforms and how frequent of Flash Loan is used in each platform.
RQ2 How frequent behaviors are launching with Flash Loan and what their distribution is in the whole blockchain ledger.
RQ3 Can ThunderStorm help security analysts to understand the speculative usage of Flash Loan in the real world?

Finally, we will demonstrate the capability of our analysis by delving into the details of the real world events in Section 5.4.

5.1 Data Set

We collect total 863,504, 142 existing transactions from Ethereum blockchain ledger with its state information, which include emitted events and invoked functions, until 10th Oct 2020.

5.2 Flash Loan Distribution

This section targets RQ1. Through running Flash-Loan Identifier on all existing transactions, 22, 244 transactions containing Flash Loan are identified. Moreover, Table 3 lists the number of Flash Loan invoked by users in each platform.

Table 3: Distribution of Flash Loan transactions.

| Providers  | # of Transactions | # of borrowers |
|-----------|-------------------|----------------|
| Aave       | 6,199             | 301            |
| bZx        | 2                 | 3              |
| UniswapV2  | 2,883             | 234            |
| dYdX       | 13,203            | 393            |
| Total      | 22,244*           | 908*           |

* As one transaction may include two Flash Loan, and it is possible for borrowers to use Flash Loan in different platforms. As a result, for each column, the sum of rows (except "Total") is more than the corresponding total number.

It can be found from the results that users leverage the Flash Loan in dYdX most frequently, as there are 13, 203 transactions which are over half of the total amount of Flash Loan transactions. Following up, 6, 199 Flash Loan transactions are found in Aave, and 2,883 Flash Loan transactions are found in UniswapV2. Note that the number in UniswapV2 could increase rapidly since its Flash Loan service is announced very recently compared to the one in dYdX and Aave. Another interesting founding is that although only 3 transactions are identified in bZx, the infamous arbitrage on 18 Feb 2020 [22] is covered within those transactions.

In Table 3, the number of borrowers is also presented. According to the results, the Flash Loan services in Aave and dYdX are more popular. Surprisingly, 60% of Flash Loan transactions are launched by only 3% of borrowers. More specifically, in Aave, the top 10 borrowers (out of 301) run nearly 70% of the Flash Loan transactions. We found that the main user of the Aave Flash Loan is DeFi Saver which takes 8 places among the top 10 borrowers. Moreover, the user who launches the second most times of Flash Loan is Frucombo platform [26] which combines all Kyber DApps to allow users to launch their arbitrage in between two DEXes: Uniswap and Kyber with their own-discovered arbitrage opportunities. In conclusion, this phenomenon reveals that Flash Loan is currently a new service which is vastly used by some large organizations, rather than individuals.

Findings #1: dYdX occupies the most (13, 202, around 60%) of Flash Loan transactions, while bZx creates the least (only 3). The proportions of Aave and UniswapV2 are 13% and 27%, respectively. Besides, the usage of Flash Loan is currently dominated by large organizations (3.5% of borrowers launch 60% transactions), rather than individuals.

5.3 Behavior Distribution

This section targets RQ2. Specifically, to understand the behaviors, we use our proposed Flash-Loan Identifier to filter 22, 244 Flash Loan transactions from all existing ones in Ethereum. Through applying both Primitive Classifier and Advanced Classifier, those filtered transactions are classified into 7 behaviors including exchange, lending & borrowing, margin trade, liquidation, arbitrage, anti-liquidation, and swap.

Table 4: Existing Behaviors.

| Behaviors            | # of Transactions |
|----------------------|-------------------|
| **Primitive**        |                   |
| Exchange             | 22,244            |
| Lending & Borrowing  | 7,767             |
| Margin Trade         | 1                 |
| Liquidation          | 164               |
| **Total**            | 22,244            |
| **Advanced**         |                   |
| Arbitrage            | 402               |
| Anti-Liqudation      | 1,871             |
| Swapping*            | 101               |
| **Total**            | 2,331             |

* There exists 55, 20 and 26 transactions performing collateral swapping, loan swapping and platform swapping respectively.

As shown in the Table 4, there are 22, 244 transactions identified by Primitive Classifier, including 22, 244, 7, 867 and 164 for exchange, lending & borrowing, and liquidation, respectively. Based on the result, we discovered that exchange, as the basic service, dominates the scope of usage in flash loans. For the margin trade, we found that only 1 transaction was conducted. After verifying this transaction, we found that it could be regarded as price manipulation [20]. This inactive behavior probably due to the leverage used therein, as a high leverage may increase the risk of collateral damage.

As for the 2,331 transactions with advanced behaviors produced via running Advanced Classifier, arbitrage, anti-liquidation, and swapping are found in 402, 1, 871, and 101 transactions, respectively. Based on the results, anti-liquidation seems to be the most

[20] i.e., a speculative user can margin trade with leverages to manipulate the price in an AMM DEX to make an opportunity for arbitrage.
Table 5: Internal transaction classification of the bZx hack.

| DeFi Behaviors                      | # Start Internal Transaction (index) | # End Internal Transaction (index) |
|-------------------------------------|-------------------------------------|-----------------------------------|
| Flash Loan in dYdX                  | 2                                   | 188                               |
| Collateral Borrowing in Compound    | 21                                  | 46                                |
| Margin Trading in bZx               | 47                                  | 174                               |
| First Swapping in Uniswap           | 158                                 | 161                               |
| Second Swapping in Uniswap          | 176                                 | 180                               |

Table 5: Internal transaction classification of the bZx hack.

Findings #2: For primitive behaviors, exchange (22, 244, 100%) dominates the scope of usage in Flash Loan, and lending & borrowing (7, 767, around 35%) takes the second place. Liquidation (164) and margin trade (1) are far more behind. For advanced behaviors, anti-liquidation associates with the most (1, 871) transactions. Arbitrage and swapping occupy 402 and 101, respectively.

5.4 Speculative Usage of Flash Loan in The Real World

This section focuses on RQ3. Specifically, DeFi ecosystem is still immature, with the new-born service Flash Loan, there exists some traders trying to gain themselves considerable profits via “smartly” leveraging the flash loan, such as infamous use cases [21] [22] [23] [24] leveraging Flash Loan. However, these cases usually cross multiple DeFi platforms, which make understanding these cases complicated. That is because they involved a lot of different contract invocation.

We take the bZx hack [2] introduced in Section 3 as an example to demonstrate that the proposed approach is capable of analyzing and understanding the speculative usage of Flash Loan. First, for each external transaction, ThunderStorm marks each internal transaction triggered by it in the order of execution (with indexes). Then, ThunderStorm records the start and end indexes of the internal transaction for each behavior when identifying DeFi behaviors in Section 4.

As shown in Table 5, the internal transactions (or contract invocations) of the bZx hack are classified automatically according to the recognized behaviors. For example, Flash Loan in dYdX (the first row in the table) has 187 internal transactions (from index 2 to index 188), which means almost all actions (e.g., Margin Trading in bZx in the third row, from 47 to index 174) must be completed before paying back this Flash Loan. With that information, we only need to manually check the contract invocation between these DeFi behaviors, and the whole scope of the bZx hack can be easily captured. By doing so, ThunderStorm will help reduce the burden for security analysts.

Figure 5 shows the attack details. In particular, the attacker launches the following five steps to complete this attack.

1. Launch the flash loan
2. Borrow 112 WBTC with 5,500 ETH as collateral
3. Launch a margin trade
4. Swap 112 WBTC for 6,871.41 ETH
5. Payback 10,000 ETH and close the flash loan

Findings #3: Based on the proposed approach, we have made a meaningful step towards understanding speculative usage of Flash Loan associated with complicated invocations and a large number of transactions.

6 DISCUSSION

In this section, we elaborate on the limitations of our work. There exist several challenges hindering from a deep understanding of behavior based merely on Ethereum transactions.

Pattern Coverage. We tried our best to collect patterns, however, the completeness of the collected patterns cannot be absolutely guaranteed. Due to the rapid iteration and security threats faced by open source and public platforms, protocols based on the agreement of the community are updated quickly to protect users’ locked capitals. However, due to such frequent updates, the pattern collection of historical smart contracts that are normally closed or destroyed becomes a challenge. Moreover, platforms only enable their development documentation with the latest version.
Interpretation of Information. To fully understand the behaviors, it requires an adequate interpretation of extracted information. Most of the extracted information can be interpreted by the full name of the collected signature. However, to understand the type of trading assets being transferred between traders and platforms, collecting corresponding flags (addresses) is compulsory to automatically launch the filtering and classifying process on a large-scale of transactions. Moreover, for instance, the leverage amount of a margin trade in bZx cannot be revealed straightly from the parameters of the key function signature. This requires us to conduct a more specific study to nail this problem.

Variety of Arbitrage. As long as two trades with the same user are identified in a transaction, we confirm that is a price difference arbitrage. To further improve the understanding of such an arbitrage pattern, however, we meet some challenges.

- Well-Connected Cash Flow. Since we only consider one user in our pattern, this leaves a question on group trading, i.e., multiple addresses are involved for arbitrage with consensus. As shown in Figure 6, we exhibit two scenarios: 1) direct transfer; and 2) transfer through an intermediate platform such as the Lending & Borrowing platform between traders. To overcome such a challenge via transfer relationship between traders, one needs to construct a cash flow based on transfer functions fixed in the ERC20 token standard or general ether transfer function.

- Profit Measurement. Accurately measuring the profit made by an arbitrage for different input and output assets requires real-time prices in each block. It is not easy to calculate the price with only state data recorded in the Ethereum ledger. Based on a prior work [49] which maintains the state for each transaction in Ethereum, one can recover the state data of involved DEXes to calculate the exchange rate to ETH for each type of asset so that the profit can be estimated based on one asset. We leave it as our future work.

7 RELATED WORK

DeFi Security and Arbitrage. Kaihua et al. [44] investigate two existing exploits which happened on 15th and 18th Feb 2020, and present the details of how traders leverage the flash loan mechanism with the trick of price manipulation to gain profit. They also proposed a process to re-boost two exploits via optimized parameters. 2.37X and 1.73X of profits gained in their simulation for two exploits, respectively. Lewis et al. [29] leverage the flash loan to execute a governance attack [54] on MakerDAO [40]. Moreover, the proposed strategy leads to a theft of 0.5B USD and unlimited mining of DAI. Bowen et al. [37] systematically investigate oracle designs of 4 DeFi platforms with open source code via comparing their price deviations, and exhibit their potential security vulnerabilities. Kamps et al. [32] aggregate the information from the existing pump and dump schemes among the classic economic, and propose a group of patterns with summarised criteria to identify potential pump and dump activities in crypto markets. Xu et al. [51] also conduct an investigation on 412 pump and dump activities to build a model that predicts the pump behavior for all assets exhibiting in DEXes by estimating its pump likelihood. Philip et al. [16] present the breadth of DEX arbitrage bots and their profit-making strategies which optimize users’ network latency and pay a high transaction gas fee to win priority gas auctions (PGAs) so that their transaction can obtain priority ordering in the competition. Through the study, Daian et al. [16] highlights that bots’ revenue far exceeds the Ethereum block reward and transaction fees, and with such high optimization fees, the blockchain consensus stability might be threatened. Eskandari et al. [20] also study the front-running issues across the top 25 most active decentralized applications (DApps) of Ethereum blockchain and exhibit the proposed solutions into useful categories.

DeFi Monetary. Clark et al. [13] give a survey about ‘stablecoin’ of DeFi protocols and comparison among a list of cryptocurrencies like Bitcoin, Ether and etc. Furthermore, Pernice et al. [43] present a comprehensive taxonomy of cryptocurrency stabilization through combining their findings and studying classical monetary policy. Moin et al. [41] systematically explore existing stablecoins via decomposing their design into various component elements and introduce their strengths, drawbacks, and future directions. Lastly, Koledner et al. [8] investigate Namecoin, which is the fork of Bitcoin, and empirically study its name services.

Smart Contract Security. Apart from the issues mentioned in DeFi ecosystem, potential or existing vulnerabilities of smart contracts, as the running base of DeFi services, could cause a considerable financial impact for DeFi as well. A group of surveys [35] [36] [5] [12] have soundly studied different aspects of the smart contract, especially in term of its vulnerabilities and security issues. In the following, we categorize the type of smart contract analysis into two classes. Static analysis, especially symbolic execution technique, has been used to examine smart contracts in different code levels: EVM bytecode [27] [33] [38], source code [31] [11] and intermediate representation [46] of smart contracts. On the other perspective, the fuzzing technique plays a vital role in dynamic analysis to detect the vulnerabilities of smart contracts. Different improved fuzzing techniques [30] [25] [50] or constructed fuzzing frameworks [28] are conducted to discover the vulnerability of smart contracts during run-time.

8 CONCLUSION

This paper takes the first step to systematically measure the financial application behind Flash Loan mechanism on the Ethereum.
With the non-collateral feature, the Flash Loan has been leveraged for advanced behaviors which match users’ original purpose such as arbitrage, preventing from liquidation or collateral swap. In this work, we present 3-phase framework named ThunderStorm to identify each behavior within the Flash Loan transactions. In total, we collect and identify 22,244 Flash Loan transactions, 22,244 primitive behavior transactions, and 2,374 advanced behavior transactions. To better understand the behaviors behind Flash Loan, we also extract the information for each behavior as well as the distribution. Finally, we use a real world case to demonstrate the capability of the proposed system.

REFERENCES

[1] 2015. ERC20 Token Standard. https://github.com/ethereum/EIPS/EIPS/eip-20.md.
[2] 2020. bZx Hack Full Disclosure (With Detailed Profit Analysis). https://medium.com/@peckshield/bzx-hack-full-disclosure-with-detailed-profit-analysis-ebf1f9b18f.
[3] 2020. bZx. Last Accessed: Sep. 2020. bZx. bzx.network/.
[4] 2015. ERC20 Token Standard. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.
[5] 2019. EOS. Last Accessed: Sep. 2019. EOS. eos.io.
[6] 2018. Ethereum. Last Accessed: Oct. 2018. Ethereum. ethereum.org.
[7] 2020. Defi prime. Last Accessed: Sep. 2020. Defi prime.com.
[8] 2019. Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshitz. 2020. Echidna: effective, usable, and fast fuzzing for smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 557–560.
[9] 2018. Giovanni Palazzi and Christian R Bennet. 2018. Assessing the vulnerability of Ethereum smart contracts. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM, 259–269.
[10] 2020. Satoshi Nakamoto et al. 2008. Bitcoin: A peer-to-peer electronic cash system. arXiv:1904.05234
[11] 2019. Jeremy Clark, Didem Demirag, and Seyedehmahsa Moosavi. 2019. SoK: Demystifying Stablecoins. In International Conference on Formal Engineering Methods. Springer, 286–304.
[12] 2017. Lewis Gudgeon, Christian R Bennet, Holger Hermanns, and Matthias Jarke. 2017. Detecting vulnerabilities in Ethereum smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, 164–186.
[13] 2018. Sanchez et al. 2018. EIP2718: Improving safety of smart contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 254–269.
[14] 2016. Bo Jiang, Ye Liu, and WK Chan. 2018. ContractFuzzer: Automatically exploit smart contracts. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 254–269.
[15] 2019. Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. arXiv:1904.05234
[16] 2020. Jeff Dingledine, Nancy van Susteren, and Ian Kuttschreuter. 2020. Analyzing safety of smart contracts. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19). 1317–1333.
[17] 2019. Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshitz. 2020. Echidna: effective, usable, and fast fuzzing for smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 557–560.
[18] 2019. Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshitz. 2020. Echidna: effective, usable, and fast fuzzing for smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 557–560.
[19] 2020. Defi Board. Last Accessed: Sep. 2020. DeFi Board. defiboard.com.
[20] 2019. Adam Back. 2019. Bitcoin: A peer-to-peer electronic cash system. arXiv:1904.05234
[21] 2020. Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshitz. 2020. Echidna: effective, usable, and fast fuzzing for smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 557–560.
[22] 2019. Lewis Gudgeon, Daniel Perez, Dominik Harz, Arthur Gervais, and Benjamin Livshitz. 2020. Echidna: effective, usable, and fast fuzzing for smart contracts. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 557–560.