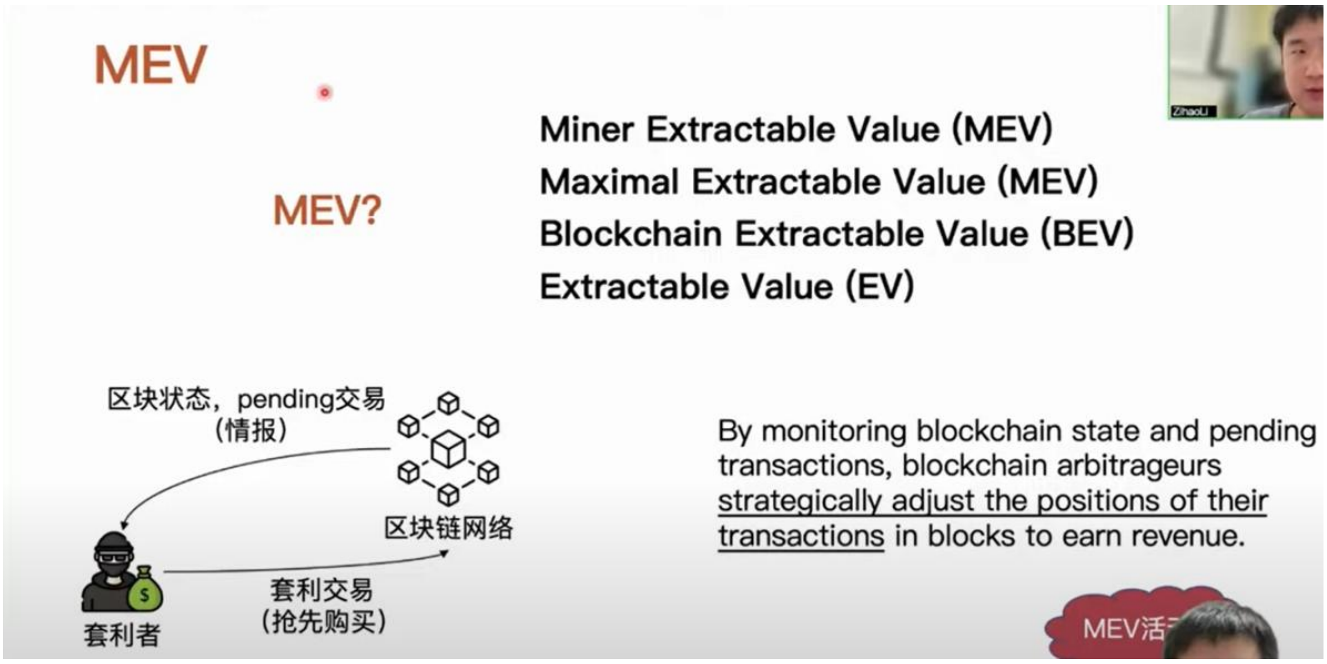
Web3青年学者计划
Web3 Young Scholars Program · Online Sharing P11
POWERED BY
DRK Lab
imToken
CRYPTAPE 秘猿科技
揭秘以太坊交易包中的MEV活动
Demystifying DeFi MEV Activities in Flashbots Bundle
预约腾讯会议
学者名片
Zihao LI
- 香港理工大学博士生，主要研究方向为区块链及智能合约安全
- 多项论文获得会议最佳论文及提名奖励，包括 ESEM 2019, INFOCOM 2018, ISPEC 2017等
Thur, Aug. 17th at 20:00pm (UTC+8)
媒体支持
学术支持
社区支持
Foresight News
PADO
SEEDAO
DappLearning
BUIDLER DAO
Rebase
SCBC
WEB3 BUIDLER

Hello everyone, I am Zihao Li, a third-year doctoral student at the Hong Kong Polytechnic University. The topic I am sharing today is 'Unveiling MEV Activities in Ethereum Transactions'. Simply put, it's about how to discover unknown types of MEV activities in the Ethereum network through transaction packages. First, I will give a basic introduction, such as the concept of MEV, transaction package mechanism, and the background of our work. Then I will introduce the complete workflow and some design ideas in detail, such as the design principles of the workflow; what datasets we have; which tools we used and how we evaluated our workflow, etc. Finally, I will introduce three applications and their empirical analysis results.

Background Introduction: MEV, Transaction Packages, and Motivation

MEV activities refer to arbitrageurs in the blockchain who generate arbitrage transactions by monitoring the blockchain network, including block states. Some transaction information is propagated in the blockchain P2P network or stored in the transaction pool of miners or validators without being officially on the chain. When an arbitrageur listens to this transaction information, he generates his own arbitrage transaction through certain strategies and then specifies the arbitrage transaction in a certain position of the next block. For example, he wants to execute a strategic transaction to propagate the same arbitrage transaction at the top of the next block or immediately after a certain transaction. Such specified arbitrage activities can be considered as MEV activities. For instance, when an arbitrageur notices a fluctuation in asset prices, he can purchase the corresponding assets in the low-priced transaction pool and then sell them at a higher price in another high-priced capital pool. We regard this as an MEV activity.

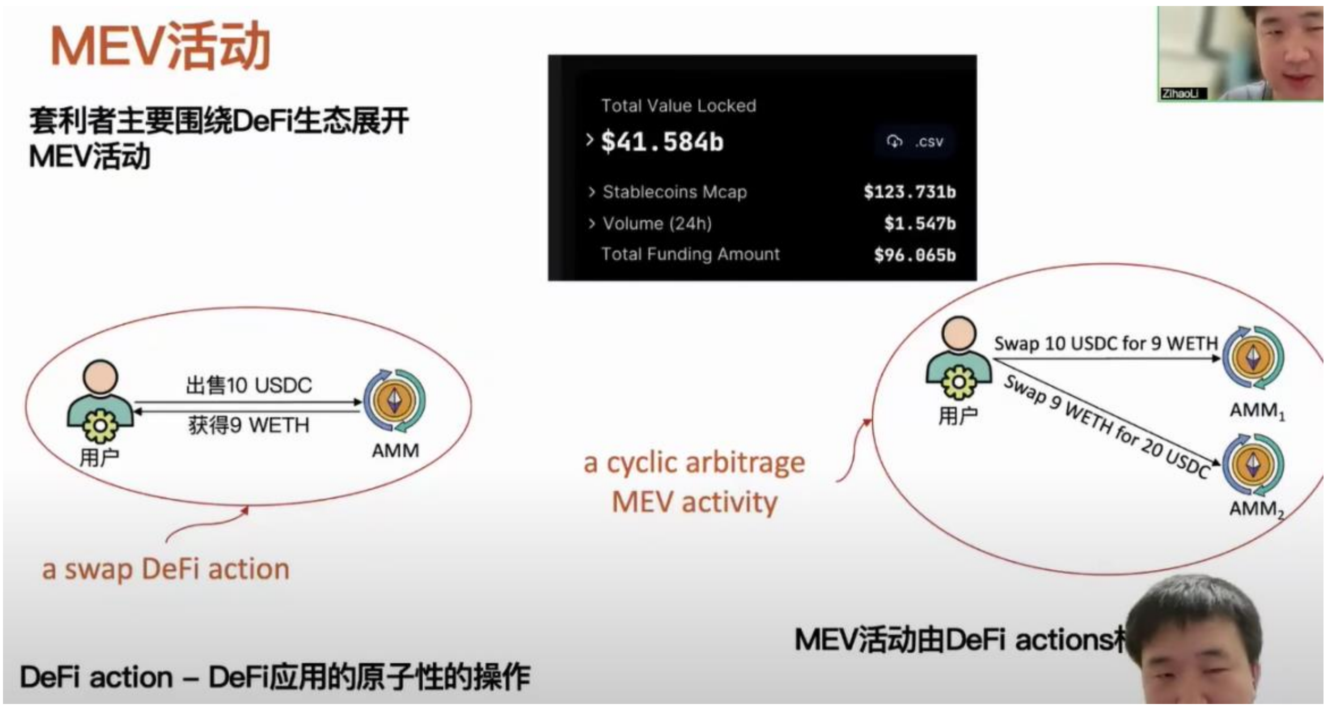


MEV

MEV?

Miner Extractable Value (MEV)
Maximal Extractable Value (MEV)
Blockchain Extractable Value (BEV)
Extractable Value (EV)

By monitoring blockchain state and pending transactions, blockchain arbitrageurs strategically adjust the positions of their transactions in blocks to earn revenue.



MEV活动

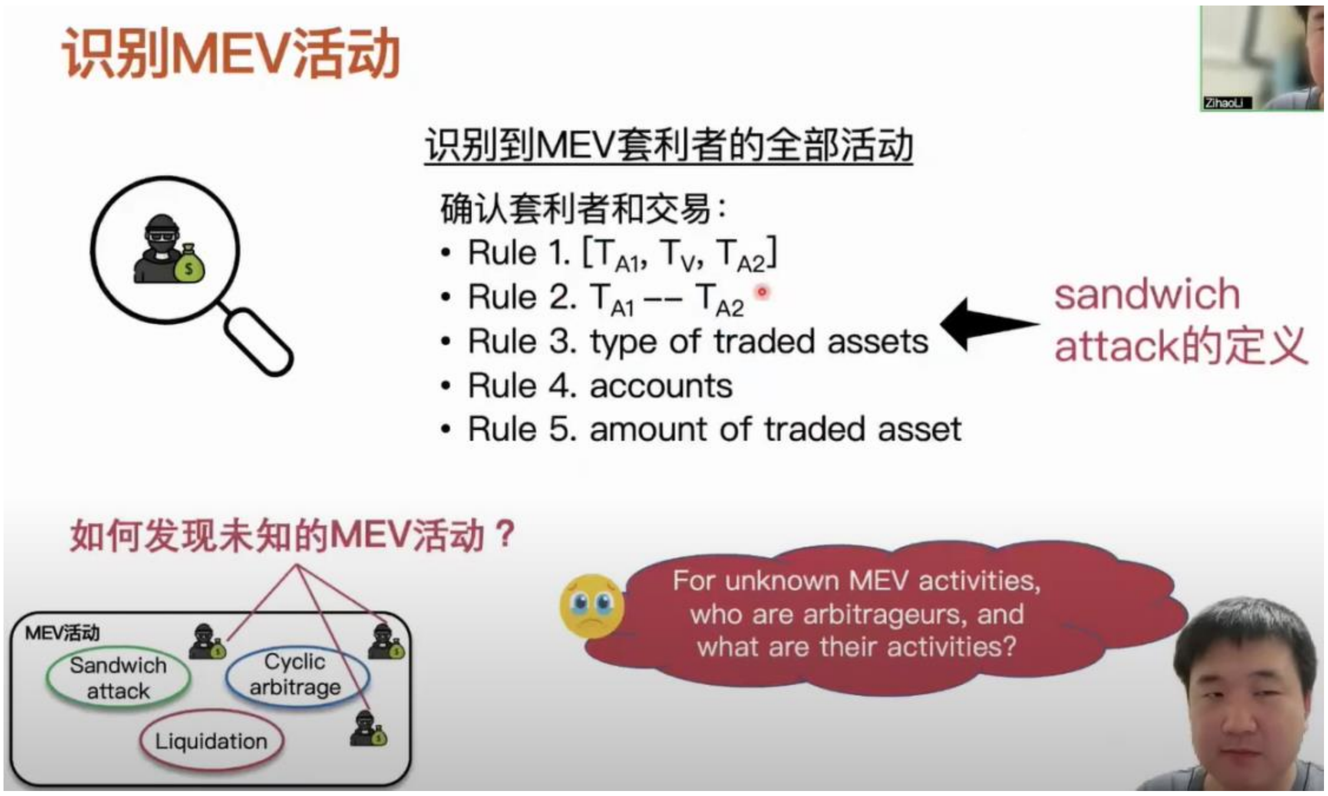
套利者主要围绕DeFi生态展开MEV活动

Total Value Locked
$41.584b
Stablecoins Mcap $123.731b
Volume (24h) $1.547b
Total Funding Amount $96.865b

DeFi action – DeFi应用的原子性的操作

MEV活动由DeFi actions

MEV activities are currently mainly centered around the DeFi ecosystem because the DeFi ecosystem primarily gathers assets. Up to now, Ethereum, including other chain's DeFi ecosystems, has attracted more than 40 billion USD in funds. Here, we need to introduce a concept about the DeFi ecosystem called "DeFi action." It corresponds to the atomic service operation provided by a DeFi application. For example, as we know, AMMs (Automated Market Makers) support exchanges between different types of assets. Users can sell some USDC and then receive ETH. Such an operation can be defined as a "DeFi action." We can use "DeFi action" to represent MEV activities. For instance, if a user detects a price discrepancy in assets across different AMMs, the user can capitalize on this difference by buying low and selling high, ultimately profiting from the price spread. We can represent this MEV activity as two DeFi actions.



识别MEV活动

识别到MEV套利者的全部活动

确认套利者和交易：

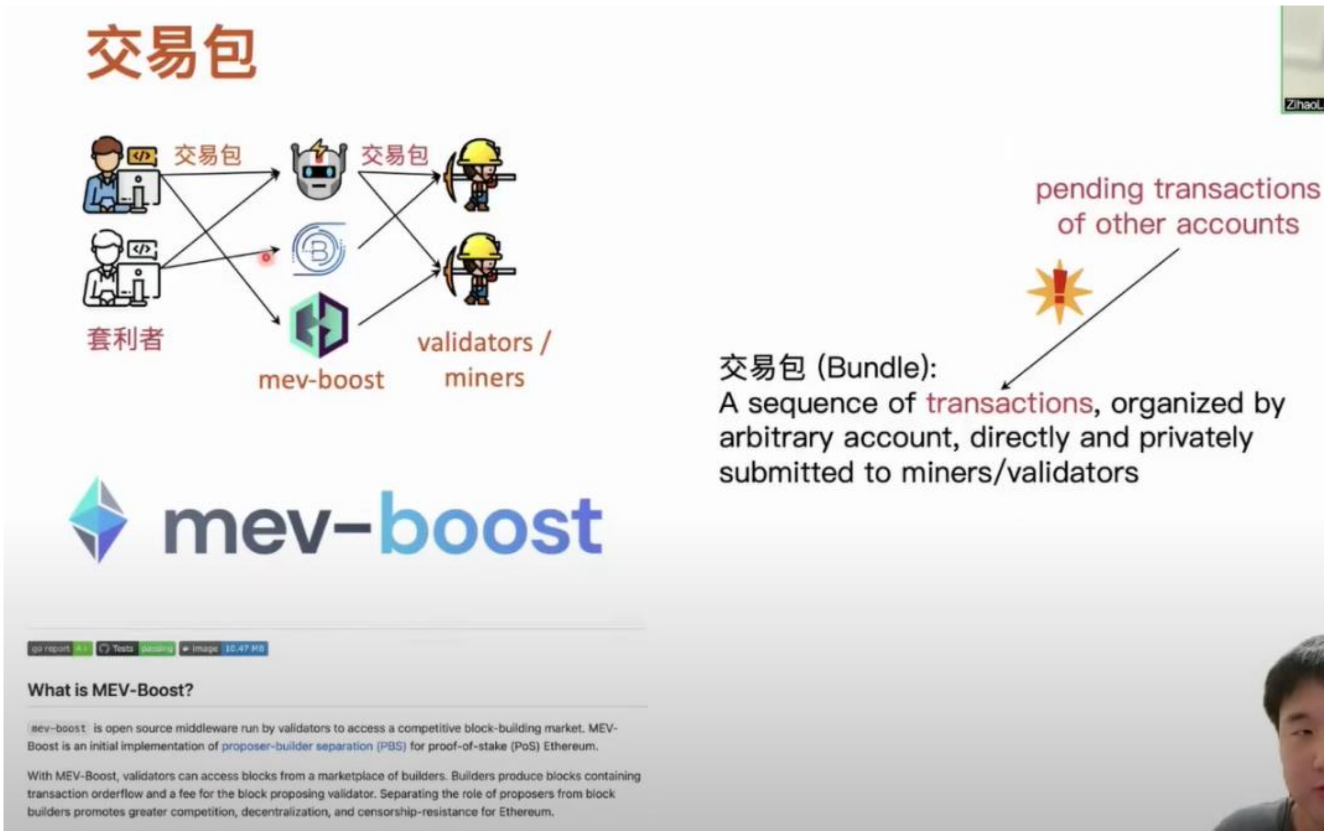
- Rule 1. $[T_{A1}, T_V, T_{A2}]$
- Rule 2. T_{A1} -- T_{A2}
- Rule 3. type of traded assets
- Rule 4. accounts
- Rule 5. amount of traded asset

sandwich attack的定义

如何发现未知的MEV活动？

For unknown MEV activities, who are arbitrageurs, and what are their activities?

Currently, academic research on MEV activities is mainly divided into three categories: sandwich attacks, reverse arbitrage, and liquidation. In our dataset, we found that these three types of MEV activities occurred more than 1 million times. There is a question here: once we know the definitions of these MEV activities, how do we recognize when they occur? If we want to identify these MEV activities, we need to recognize all the activities of the arbitrageurs. For example, which transactions do arbitrageurs generate, and what types of arbitrages are in these transactions? Only then can we determine which type of MEV activity is currently happening, and the whole process heavily relies on our definition of known MEV activities. Taking the sandwich attack as an example, once we know its definition and want to determine its arbitrage value and corresponding arbitrage transaction, we need to set many rules based on the definition. Then, we can filter out the candidate sandwich attack arbitrage values and transactions through these rules. When identifying known MEV attack types in this way, there are two problems. The first question is: can the three known common MEV activities represent all MEV activities? Clearly not, because the DeFi ecosystem is continuously evolving, new applications are constantly being developed, and the strategies of these arbitrageurs themselves are also continuously iterating. The second question is: how can we discover these unknown MEV activities? We consider this question while examining the transaction package mechanism.



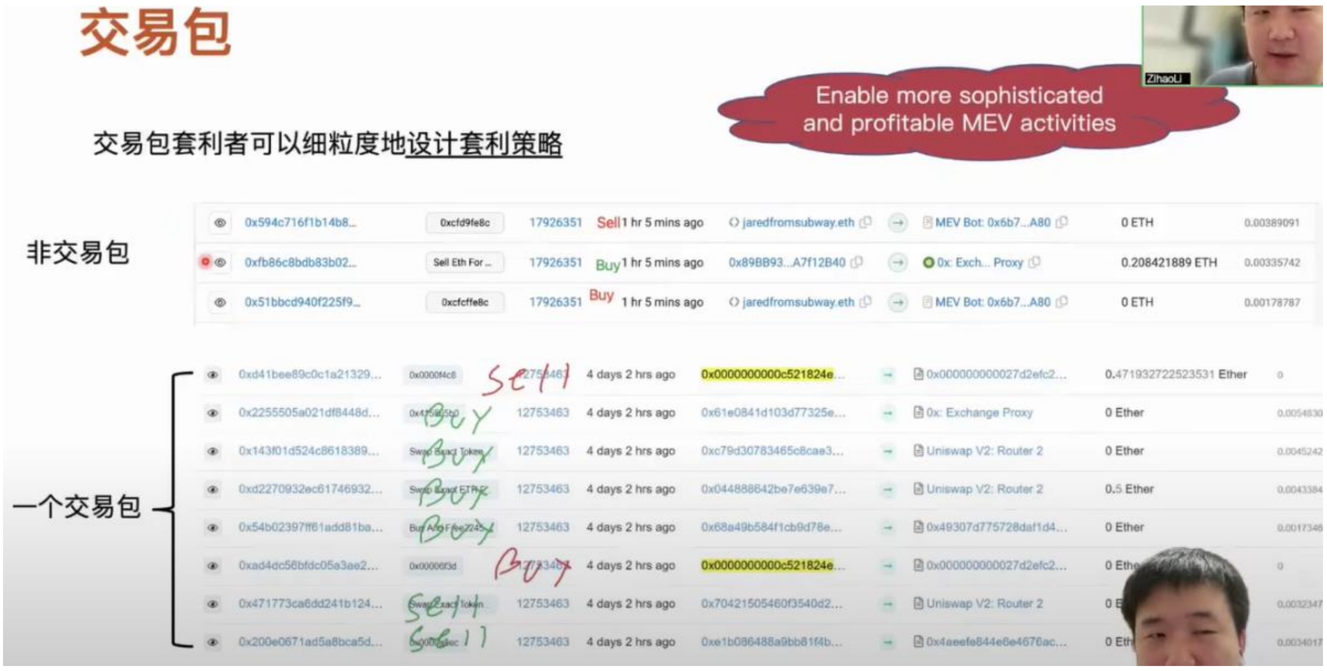
交易包 (Bundle):
A sequence of transactions, organized by arbitrary account, directly and privately submitted to miners/validators

What is MEV-Boost?

`mev-boost` is open source middleware run by validators to access a competitive block-building market. MEV-Boost is an initial implementation of proposer-builder separation (PBS) for proof-of-stake (PoS) Ethereum.

With MEV-Boost, validators can access blocks from a marketplace of builders. Builders produce blocks containing transaction orderflow and a fee for the block proposing validator. Separating the role of proposers from block builders promotes greater competition, decentralization, and censorship-resistance for Ethereum.

The transaction package mechanism was first proposed in 2021. In simple terms, users can organize a transaction queue. The length of this transaction queue can be a single transaction or several transactions. Users then send these transactions to relayers in the blockchain network. After collecting these transactions, the relayers directly and privately send them to the relevant miners or validators. Currently, relayers run transaction packages to undertake relay tasks. An essential feature of the transaction package mechanism is that when users construct a transaction package, they can include transactions from others that have not yet been added to the chain, and the order of transactions within the package can be manipulated at will. At this point, the user of the transaction package, or the arbitrageur using the transaction package, can design their arbitrage rules. For instance, they can design more complex and more profitable MEV activity strategies. Taking the sandwich attack as an example, without using a transaction package, an arbitrageur of a sandwich attack would need to generate at least one pair of transactions to realize the arbitrage, and this pair of arbitrage transactions can only target one transaction. The arbitrage from this attack transaction must be executed in a certain order to ensure its success. However, if an arbitrageur uses a transaction package, they can collect many arbitrable transactions, and with just one corresponding pair of arbitrage transactions, they can arbitrage multiple transactions simultaneously. As long as this transaction package is added to the chain, it is guaranteed to be successful in its arbitrage. Furthermore, because it arbitrages multiple transactions at once, its results yield higher profits.



交易包

交易包套利者可以细粒度地设计套利策略

Enable more sophisticated and profitable MEV activities

非交易包

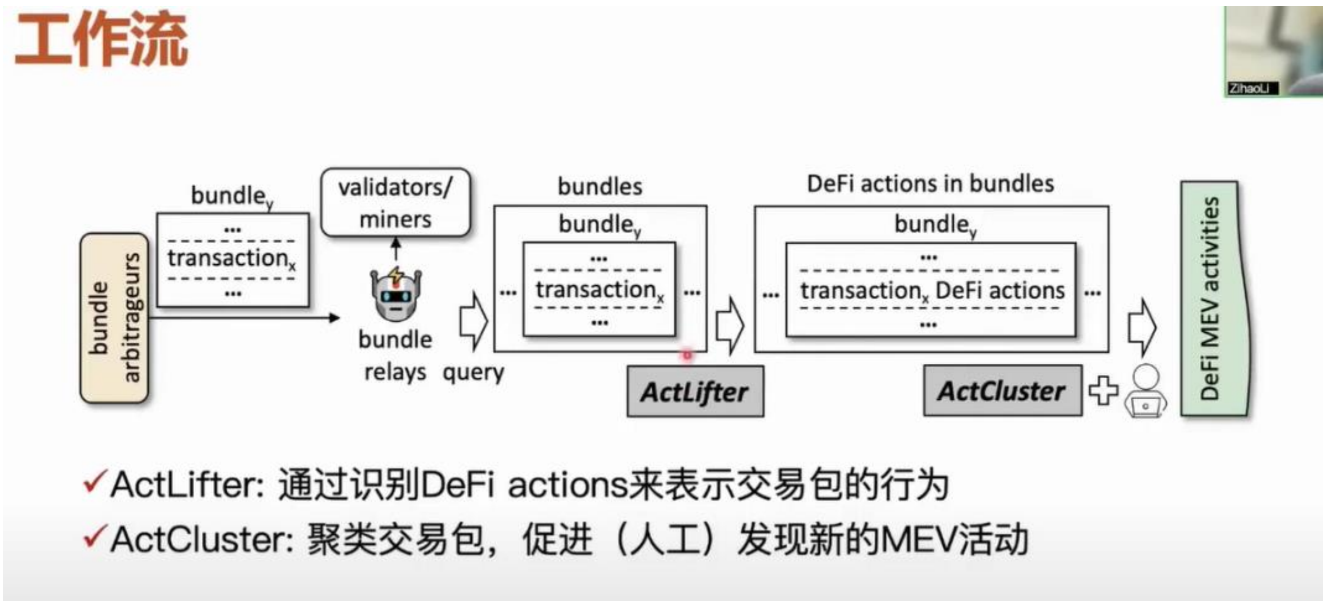
0x594c716f1b14b8...	0xcfd9fe8c	17926351	Sell 1 hr 5 mins ago	jaredfromsubway.eth	MEV Bot: 0x6b7...A80	0 ETH	0.00389091
0xfb86c8bdb83b02...	Sell Eth For ...	17926351	Buy 1 hr 5 mins ago	0x89BB93...A7f12B40	0x: Exch... Proxy	0.208421889 ETH	0.00335742
0x51bbcd940f225f9...	0xcfcffe8c	17926351	Buy 1 hr 5 mins ago	jaredfromsubway.eth	MEV Bot: 0x6b7...A80	0 ETH	0.00178787

一个交易包

0xd41bee89c0c1a21329...	0x0000f4c8 Sell	[illegible]	4 days 2 hrs ago	0x0000000000c521824e...	0x000000000027d2efc2...	0.471932722523531 Ether	0
0x2255505a021df8448d...	[illegible] Buy	12753463	4 days 2 hrs ago	0x61e0841d103d77325e...	0x: Exchange Proxy	0 Ether	0.0054830
0x143f01d524c8618389...	Swap Exact Token Buy	12753463	4 days 2 hrs ago	0xc79d30783465c8cae3...	Uniswap V2: Router 2	0 Ether	0.0045242
0xd2270932ec61746932...	Swap Exact ETH F Buy	12753463	4 days 2 hrs ago	0x044888642be7e639e7...	Uniswap V2: Router 2	0.5 Ether	0.0043384
0x54b02397ff61add81ba...	Buy And Free2245 Buy	12753463	4 days 2 hrs ago	0x68a49b584f1cb9d78e...	0x49307d775728daf1d4...	0 Ether	0.0017346
0xad4dc56bfdc05a3ae2...	0x00006f3d Buy	[illegible]	4 days 2 hrs ago	0x0000000000c521824e...	0x000000000027d2efc2...	0 Ether	0
0x471773ca6dd241b124...	Swap Exact Token Sell	12753463	4 days 2 hrs ago	0x70421505460f3540d2...	Uniswap V2: Router 2	0 E	0.0032347
0x200e0671ad5a8bca5d...	[illegible] Sell	12753463	4 days 2 hrs ago	0xe1b086488a9bb81f4b...	0x4aeefe844e6e4676ac...	0 Eth	0.0034017

The characteristic of transaction packages is that they have very rich and complex MEV activities. Because users using transaction packages encapsulate their complete transactions within the package, they then send it to relayers in the P2P network, and finally to the corresponding miners and validators. Through transaction packages, we can accurately and comprehensively identify all activities. Therefore, we can more precisely identify some unknown MEV activities through the medium of transaction packages.

Workflow and Design Ideas

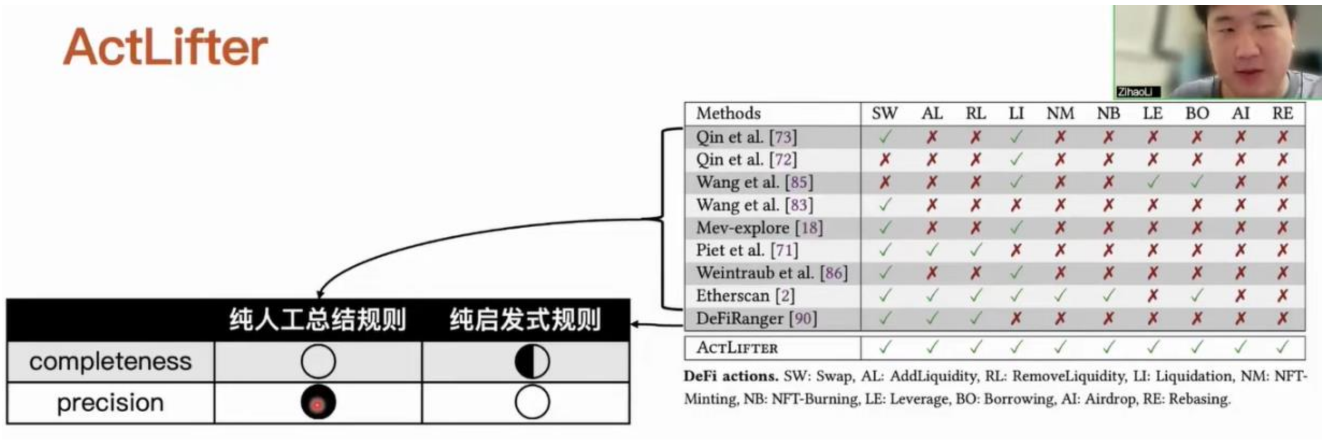


工作流

✓ActLifter: 通过识别DeFi actions来表示交易包的行为

✓ActCluster: 聚类交易包，促进（人工）发现新的MEV活动

Next, let's delve into our workflow. How do we discover unknown MEV activities through the medium of transaction packages? The core workflow includes two tools. First, after the relayer collects the transaction package, we use the ActLifter tool to identify each DeFi action in the transaction package. Once we have the results, we then represent all behaviors within the transaction package. We then employ the ActCluster tool, which uses a clustering method to group transaction packages with similar activities together. By clustering the results, we can more rapidly discover new MEV activities. If we want to uncover unknown MEV activities, manual verification to confirm whether an MEV activity is of an unknown type is inevitably required. Of course, the objective of our design is to minimize manual workload as much as possible and to make the entire process as automated as possible.


ActLifter

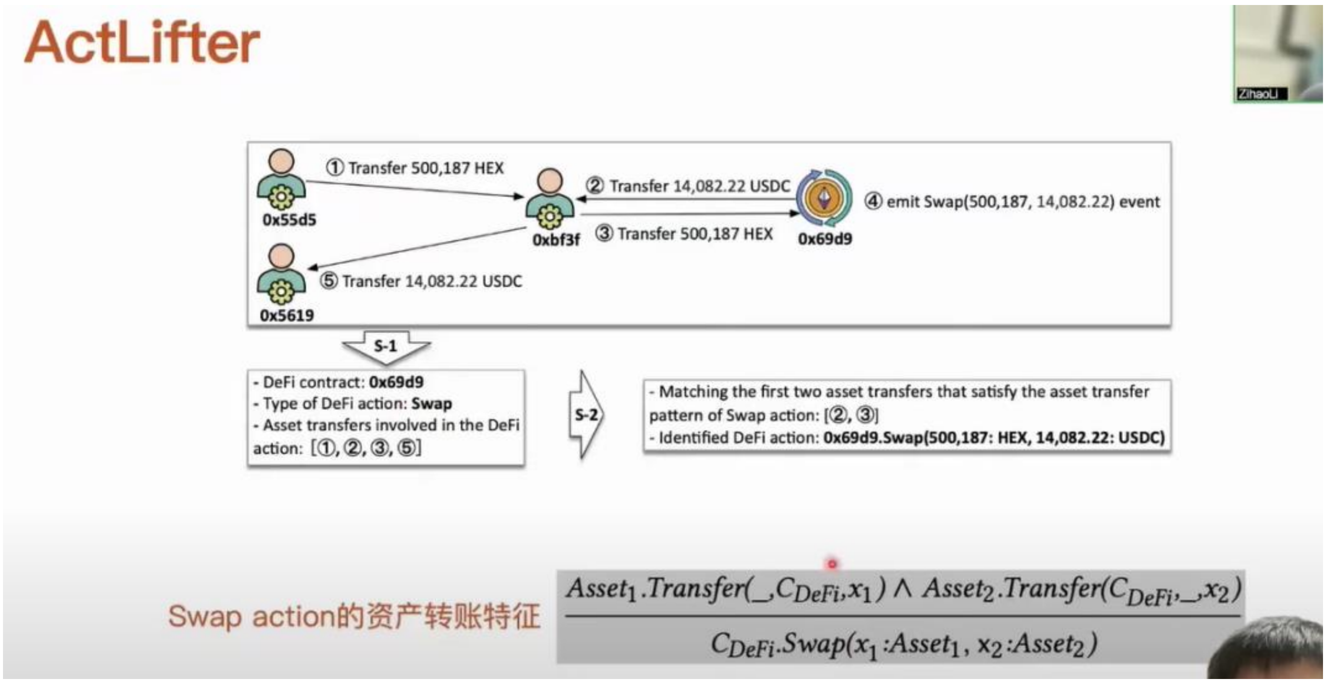
	纯人工总结规则	纯启发式规则
completeness	○	◐
precision	●	○

Methods	SW	AL	RL	LI	NM	NB	LE	BO	AI	RE
Qin et al. [73]	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Qin et al. [72]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Wang et al. [85]	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗
Wang et al. [83]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Mev-explore [18]	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Piet et al. [71]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
Weintraub et al. [86]	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Etherscan [2]	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗
DeFiRanger [90]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
ActLifter	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

DeFi actions. SW: Swap, AL: AddLiquidity, RL: RemoveLiquidity, LI: Liquidation, NM: NFT-Minting, NB: NFT-Burning, LE: Leverage, BO: Borrowing, AI: Airdrop, RE: Rebasing.

Currently, there are some tools that can identify MEV activities from transactions. Roughly speaking, they can be divided into two categories. The first type involves purely manual rule summarization, while the second is purely heuristic rules — that is, using a fully automated heuristic rule to identify specific types of MEV activities. For instance, after it recognizes some current transfer information, it checks if the heuristic rule is satisfied. If met, the corresponding activity can be identified. The first method, which is entirely based on manual rule summarization, can achieve good accuracy because the process involves thorough manual analysis of specific applications. As a result, it can ensure that the detection results are accurate. However, the analysis task involves a substantial workload, meaning it cannot cover every DeFi application. The second method, although fully automated, can only cover specific types. Additionally, there are some issues in the design of heuristic rules, resulting in unsatisfactory recognition accuracy.

Combining the advantages of the two methods, we designed our workflow. We can identify ten of the main DeFi actions currently in use. We only need to manually determine which event in the DeFi application corresponds to which type of DeFi action. After that, we don't require manual analysis, and the rest can be fully automated. The second method can entirely automatically recognize DeFi actions, but it cannot determine if the analyzed object is related to MEV activities. For instance, if we recognize a SWAP transfer, it might combine two entirely unrelated transfers to identify as a single DeFi action, naturally resulting in an incorrect identification. However, we can use this information to filter out genuinely relevant DeFi action-related information. Once we have this information, we can avoid some of the errors that occur in the second method through automated means.


ZihaoLi
① Transfer 500,187 HEX
0x55d5
0xbf3f
② Transfer 14,082.22 USDC
③ Transfer 500,187 HEX
0x69d9
④ emit Swap(500,187, 14,082.22) event
⑤ Transfer 14,082.22 USDC
0x5619
S-1
S-2

ActLifter

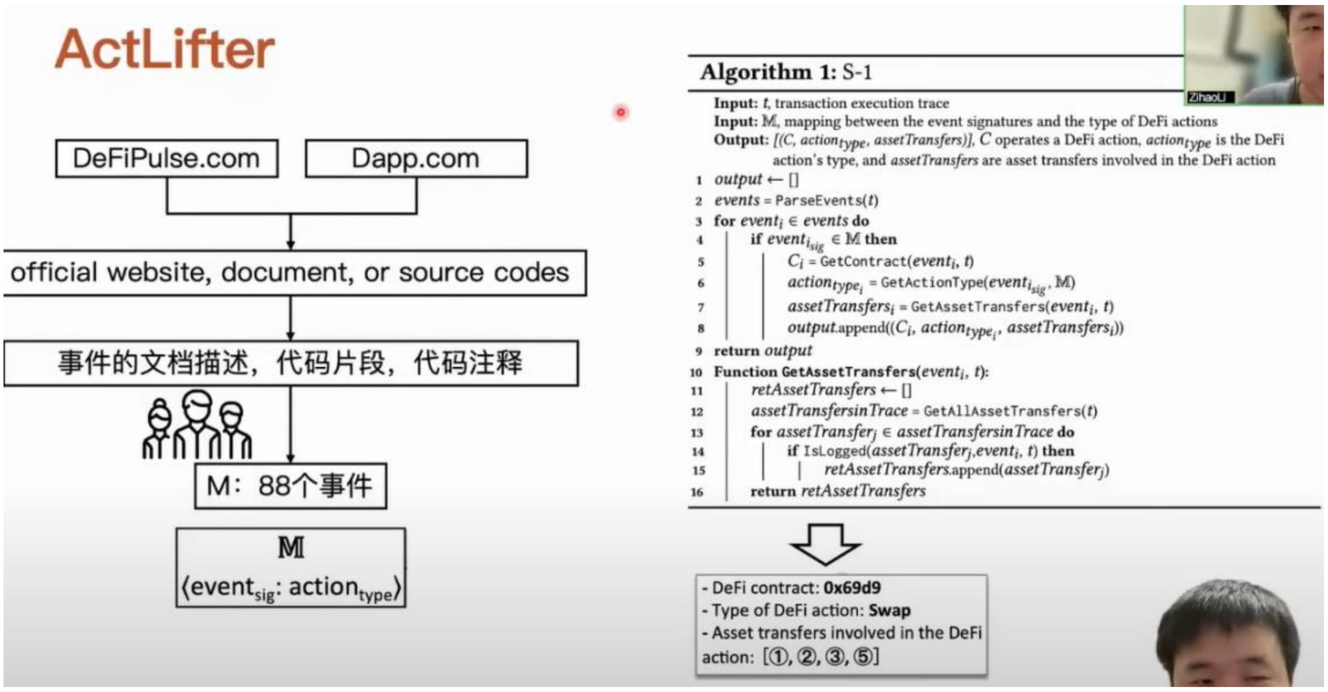
- DeFi contract: **0x69d9**
- Type of DeFi action: **Swap**
- Asset transfers involved in the DeFi action: [①, ②, ③, ⑤]

- Matching the first two asset transfers that satisfy the asset transfer pattern of Swap action: [②, ③]
- Identified DeFi action: **0x69d9.Swap(500,187: HEX, 14,082.22: USDC)**

Swap action的资产转账特征

$$\frac{Asset_1.Transfer(\_, C_{DeFi}, x_1) \land Asset_2.Transfer(C_{DeFi}, \_, x_2)}{C_{DeFi}.Swap(x_1 : Asset_1, x_2 : Asset_2)}$$

For instance, suppose there's a transaction that involves four transfers. Their sequence, amount, type, etc., are indicated by serial numbers. During this process, the AMM initiated an event related to a Swap action. With the first method, once this event is determined, it recovers the current content by looking at some parameters of the event. For example, it needs to examine the code of contract 699, its business logic, and some function variables to recover the current content. After obtaining this information, based on the unique asset transfer characteristics, we designed a rule. For instance, the rule we extracted is that the current DeFi contract has received and sent out different types of assets. When we find that two such asset transfers match these characteristics, we can recover the content of the corresponding Swap action. The second method directly matches two asset transfers, where the accounts received and transferred different types of assets. The first and fifth transfers are considered by it to be a pair of related transfers, and it believes that the intermediary account is an AMM. Clearly, as we can intuitively see, the recognition result is inaccurate.


ActLifter
DeFiPulse.com
Dapp.com
official website, document, or source codes
事件的文档描述，代码片段，代码注释
M：88个事件
$\mathbb{M}$
$\langle event_{sig}: action_{type} \rangle$
ZihaoLi

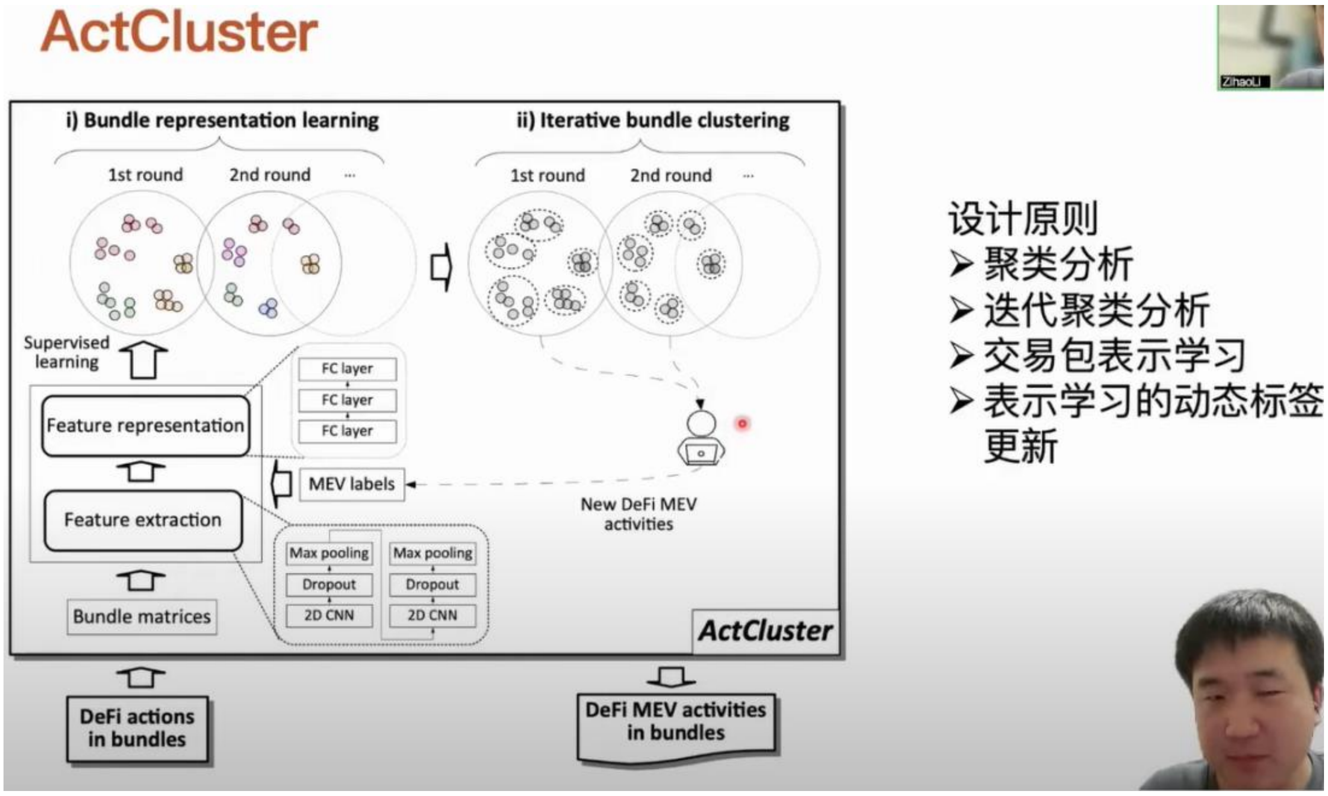
Algorithm 1: S-1

```
Input: $t$, transaction execution trace
Input: $\mathbb{M}$, mapping between the event signatures and the type of DeFi actions
Output: $[(C, action_{type}, assetTransfers)]$, $C$ operates a DeFi action, $action_{type}$ is the DeFi
        action's type, and $assetTransfers$ are asset transfers involved in the DeFi action
$output \leftarrow []$
$events$ = ParseEvents($t$)
for $event_i \in events$ do
    if $event_{i_{sig}} \in \mathbb{M}$ then
        $C_i$ = GetContract($event_i$, $t$)
        $action_{type_i}$ = GetActionType($event_{i_{sig}}$, $\mathbb{M}$)
        $assetTransfers_i$ = GetAssetTransfers($event_i$, $t$)
        $output$.append(($C_i$, $action_{type_i}$, $assetTransfers_i$))
return $output$
Function GetAssetTransfers($event_i$, $t$):
    $retAssetTransfers \leftarrow []$
    $assetTransfersinTrace$ = GetAllAssetTransfers($t$)
    for $assetTransfer_j \in assetTransfersinTrace$ do
        if IsLogged($assetTransfer_j$, $event_i$, $t$) then
            $retAssetTransfers$.append($assetTransfer_j$)
    return $retAssetTransfers$
```

- DeFi contract: **0x69d9**
- Type of DeFi action: **Swap**
- Asset transfers involved in the DeFi action: [①, ②, ③, ⑤]

The rules we derived through manual analysis correspond to the types of DeFi actions related to the events. Although the results were obtained through manual analysis, we still tried to refine the manual analysis process into a semi-automated one, ensuring the reliability of the entire procedure. We would query the official websites of DeFi applications, developer documentation, and some contract source codes from DeFiPulse.com and Dapp.com. The parsing tools we developed can extract descriptions about events from these materials, such as how the event is defined using tokens, in which functions, and the code snippets and comments where these events are used. After extracting this information and discussing it through our manual analysis, we eventually determined that there are 88 events, each corresponding to different types of DeFi actions.

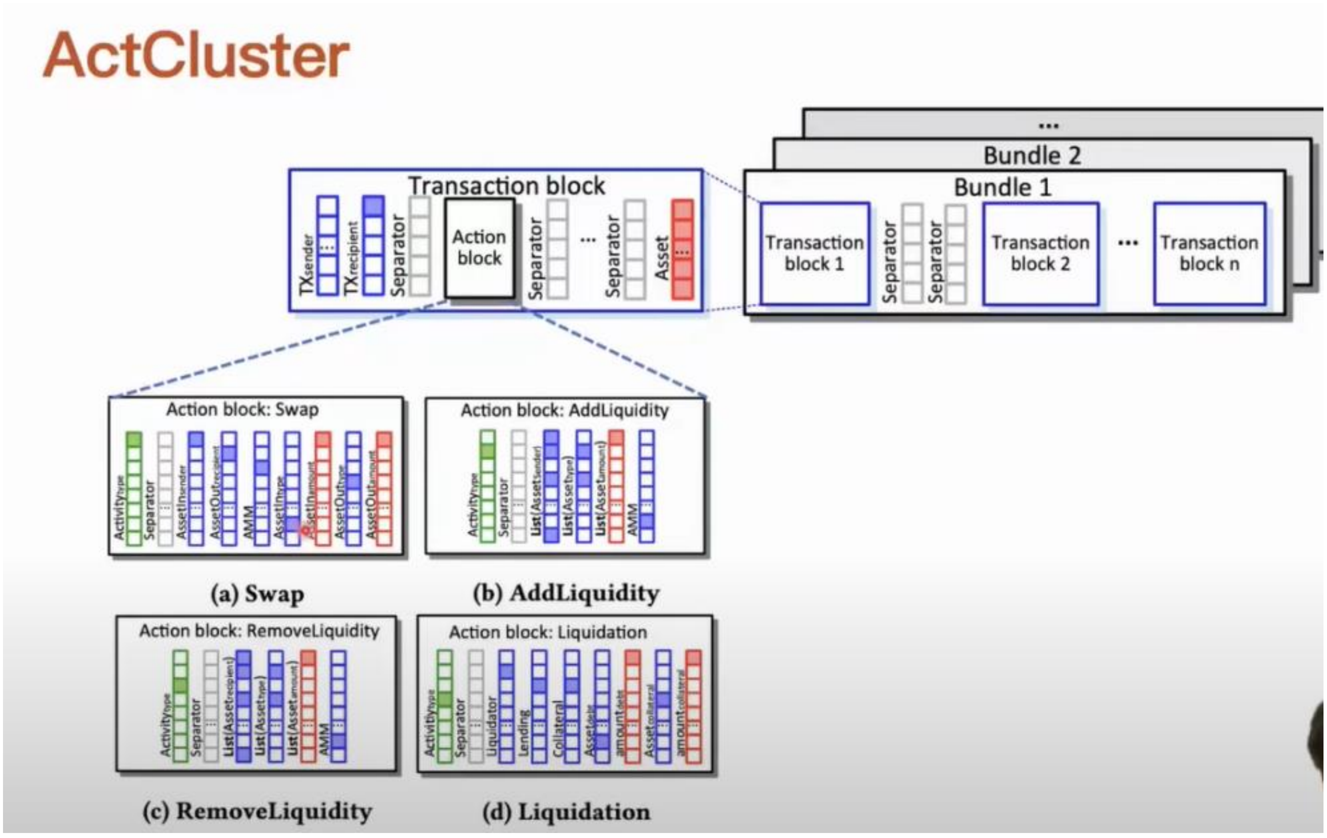
We input the transactions to be analyzed into this dictionary and parse which events occurred from the transaction. Then, when an event appears in this dictionary, we extract key information according to the corresponding rules, such as which contract is operating this DeFi action, what type of DeFi it is, and which asset transfers are related to this DeFi action. After obtaining such content, we summarize the characteristic rules of asset transfers and then use these rules to match the final DeFi action. Starting from the definition of ten DeFi actions, we summarized the characteristic rules of asset transfers. After collecting this information in the previous step, we use these matching rules to match, ultimately helping us identify what specific content of which DeFi occurred in this transaction. After ActCluster identifies every transaction in the transaction package, we can then represent the behavior of the transaction package.



设计原则
- 聚类分析
- 迭代聚类分析
- 交易包表示学习
- 表示学习的动态标签更新

First, let's understand the design principles of ActCluster. We know that manual analysis in this process is inevitable. It must rely on human judgment to determine whether the activity in the transaction package is an unknown type of MEV activity. Based on this, our basic approach is to cluster similar transaction packages together through clustering. For each cluster, we only need to randomly sample one or a few transaction packages for analysis, which accelerates the manual analysis process, ultimately discovering different types of MEV activities. When we use cluster analysis to cluster transaction packages, we face a dilemma. When we set the clustering granularity coarser, transaction packages containing different types of activities will be clustered together. Although the number of clusters is reduced, and the corresponding manual analysis tasks are also reduced, some new MEV activities might be missed. If we set the clustering granularity finer, though we can distinguish some similar but different MEV activities corresponding to transaction packages, the workload for manual analysis greatly increases.

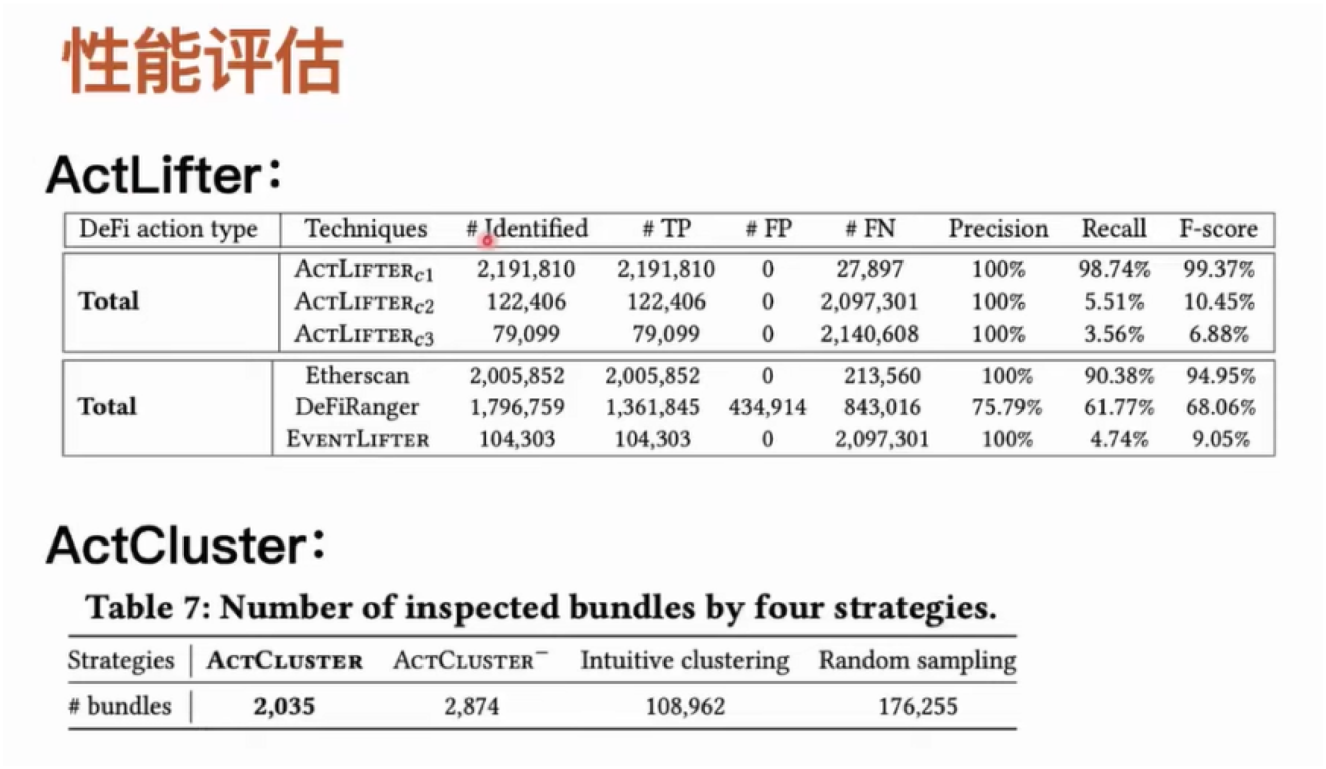
Given this issue, we designed an iterative cluster analysis method that undergoes multiple rounds of iterative clustering. In each round, we exclude transaction packages containing MEV activities discovered in the previous rounds and then increase the clustering granularity for the remaining transaction packages. We can't directly use traditional clustering methods for transaction packages because a transaction package contains multiple transactions, and a transaction can have multiple DeFi actions. If we represent the entire transaction package, its structure is heterogeneous and layered. At this point, we use representation learning to map the content of the transaction package into a fixed space. The advantage of using representation learning is that we don't need to deeply learn and understand the data we're analyzing or need extensive domain knowledge; we can simply perform data-driven processing.



For instance, we only need to label the transaction packages, indicating which MEV activities are contained within. If we know the definition of an MEV activity, it's relatively easy to design corresponding rules to automatically detect its presence. We can automate the labeling of transaction packages intended for representation learning. Our cluster analysis is iterative, and after each iteration, we can discover new MEV activities. At this point, we can enrich the labels corresponding to these newly discovered MEV activities into our representation learning process. As the labels used in representation learning are enriched, the performance and efficiency of the entire representation learning model can be iteratively improved. Furthermore, the ability of representation learning to represent transaction package activities can also be iteratively enhanced.

A transaction package can contain multiple transactions, and a transaction can also have multiple DeFi actions. We need to represent the transaction package accordingly. First, for each type of DeFi action, we define a standardized parameter, such as which contract is operating, and the quantity and type of assets received and sent. We define each DeFi action in this way. If we recognize multiple DeFi actions in a transaction, we represent them using an "action block", allowing us to represent the corresponding "transaction block" of the transaction, which includes the transaction's source information, such as who initiated it and who it's being sent to. For the DeFi actions occurring in the transaction, we fill in the action blocks in sequence. Each transaction is represented by a transaction block, and ultimately, we obtain the structure of the transaction package, which can be considered a matrix. After representing the transaction package, we can proceed with representation learning. Each transaction package has a uniform structure, allowing us to process them in batches using the model.

Performance Evaluation



性能评估

ActLifter：

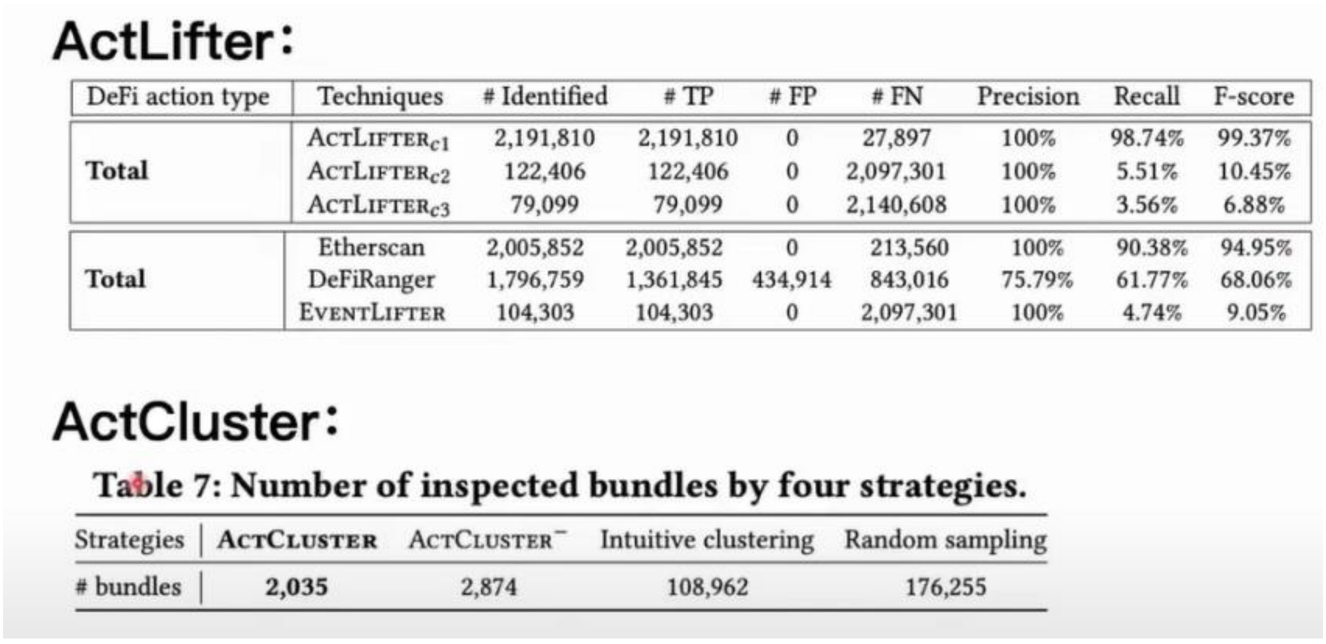
DeFi action type	Techniques	# Identified	# TP	# FP	# FN	Precision	Recall	F-score
Total	ActLifter$_{c1}$	2,191,810	2,191,810	0	27,897	100%	98.74%	99.37%
	ActLifter$_{c2}$	122,406	122,406	0	2,097,301	100%	5.51%	10.45%
	ActLifter$_{c3}$	79,099	79,099	0	2,140,608	100%	3.56%	6.88%
Total	Etherscan	2,005,852	2,005,852	0	213,560	100%	90.38%	94.95%
	DeFiRanger	1,796,759	1,361,845	434,914	843,016	75.79%	61.77%	68.06%
	EventLifter	104,303	104,303	0	2,097,301	100%	4.74%	9.05%

ActCluster：

Table 7: Number of inspected bundles by four strategies.

Strategies	ActCluster	ActCluster^{-}	Intuitive clustering	Random sampling
# bundles	**2,035**	2,874	108,962	176,255

Next, I'll share the methods we used to evaluate the performance of our workflow. The dataset for our entire analysis process was obtained through the Flashbots API, collecting transaction package data from February 2021 to December 2022, comprising over 6 million transaction packages and 26 million transactions.



ActLifter：

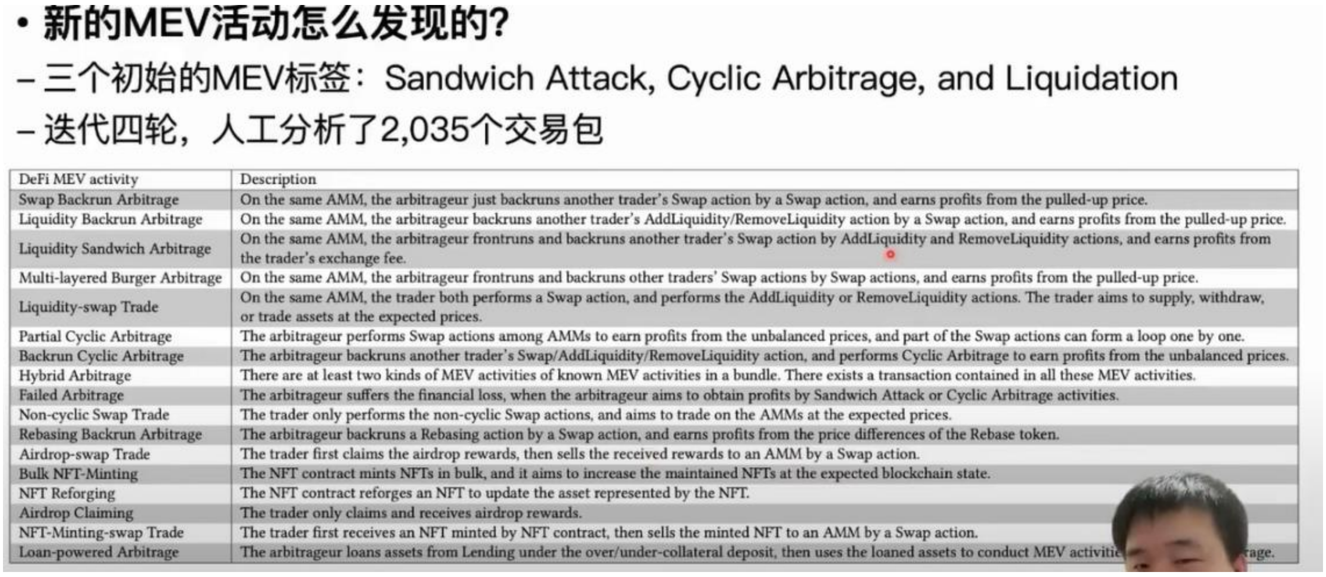
DeFi action type	Techniques	# Identified	# TP	# FP	# FN	Precision	Recall	F-score
Total	ActLifter$_{c1}$	2,191,810	2,191,810	0	27,897	100%	98.74%	99.37%
	ActLifter$_{c2}$	122,406	122,406	0	2,097,301	100%	5.51%	10.45%
	ActLifter$_{c3}$	79,099	79,099	0	2,140,608	100%	3.56%	6.88%
Total	Etherscan	2,005,852	2,005,852	0	213,560	100%	90.38%	94.95%
	DeFiRanger	1,796,759	1,361,845	434,914	843,016	75.79%	61.77%	68.06%
	EventLifter	104,303	104,303	0	2,097,301	100%	4.74%	9.05%

ActCluster：

Table 7: Number of inspected bundles by four strategies.

Strategies	ActCluster	ActCluster^{-}	Intuitive clustering	Random sampling
# bundles	**2,035**	2,874	108,962	176,255

We designed tools to compare the precision and completeness of DeFi actions. Note that, among the tools available for these chains, only Etherscan can recover DeFi actions from transactions through its website and the information it provides. For tools like DeFiRanger, we reproduced their methods based on their published papers. Additionally, we designed a tool called EventLifter, attempting to recover DeFi actions directly from transaction events. We tested ActLifter under various configurations and compared its precision using different tools. For ActCluster, our main approach was to use ablation studies. For the new activities we could identify, after ablating parts of ActCluster, if we wanted to recognize some undiscovered new activities, we needed to determine how many transaction packages required manual analysis or how extensive the manual analysis workload would be. For example, after ablating the dynamic label update in the representation learning module of ActCluster, we essentially removed the entire process. We randomly sampled from the 6 million transaction packages to see how many we'd have to manually analyze to discover the same number of new MEV activities.



- **新的MEV活动怎么发现的?**
  - 三个初始的MEV标签：Sandwich Attack, Cyclic Arbitrage, and Liquidation
  - 迭代四轮，人工分析了2,035个交易包

DeFi MEV activity	Description
Swap Backrun Arbitrage	On the same AMM, the arbitrageur just backruns another trader's Swap action by a Swap action, and earns profits from the pulled-up price.
Liquidity Backrun Arbitrage	On the same AMM, the arbitrageur backruns another trader's AddLiquidity/RemoveLiquidity action by a Swap action, and earns profits from the pulled-up price.
Liquidity Sandwich Arbitrage	On the same AMM, the arbitrageur frontruns and backruns another trader's Swap action by AddLiquidity and RemoveLiquidity actions, and earns profits from the trader's exchange fee.
Multi-layered Burger Arbitrage	On the same AMM, the arbitrageur frontruns and backruns other traders' Swap actions by Swap actions, and earns profits from the pulled-up price.
Liquidity-swap Trade	On the same AMM, the trader both performs a Swap action, and performs the AddLiquidity or RemoveLiquidity actions. The trader aims to supply, withdraw, or trade assets at the expected prices.
Partial Cyclic Arbitrage	The arbitrageur performs Swap actions among AMMs to earn profits from the unbalanced prices, and part of the Swap actions can form a loop one by one.
Backrun Cyclic Arbitrage	The arbitrageur backruns another trader's Swap/AddLiquidity/RemoveLiquidity action, and performs Cyclic Arbitrage to earn profits from the unbalanced prices.
Hybrid Arbitrage	There are at least two kinds of MEV activities of known MEV activities in a bundle. There exists a transaction contained in all these MEV activities.
Failed Arbitrage	The arbitrageur suffers the financial loss, when the arbitrageur aims to obtain profits by Sandwich Attack or Cyclic Arbitrage activities.
Non-cyclic Swap Trade	The trader only performs the non-cyclic Swap actions, and aims to trade on the AMMs at the expected prices.
Rebasing Backrun Arbitrage	The arbitrageur backruns a Rebasing action by a Swap action, and earns profits from the price differences of the Rebase token.
Airdrop-swap Trade	The trader first claims the airdrop rewards, then sells the received rewards to an AMM by a Swap action.
Bulk NFT-Minting	The NFT contract mints NFTs in bulk, and it aims to increase the maintained NFTs at the expected blockchain state.
NFT Reforging	The NFT contract reforges an NFT to update the asset represented by the NFT.
Airdrop Claiming	The trader only claims and receives airdrop rewards.
NFT-Minting-swap Trade	The trader first receives an NFT minted by NFT contract, then sells the minted NFT to an AMM by a Swap action.
Loan-powered Arbitrage	The arbitrageur loans assets from Lending under the over/under-collateral deposit, then uses the loaned assets to conduct MEV activiti [illegible] rage.

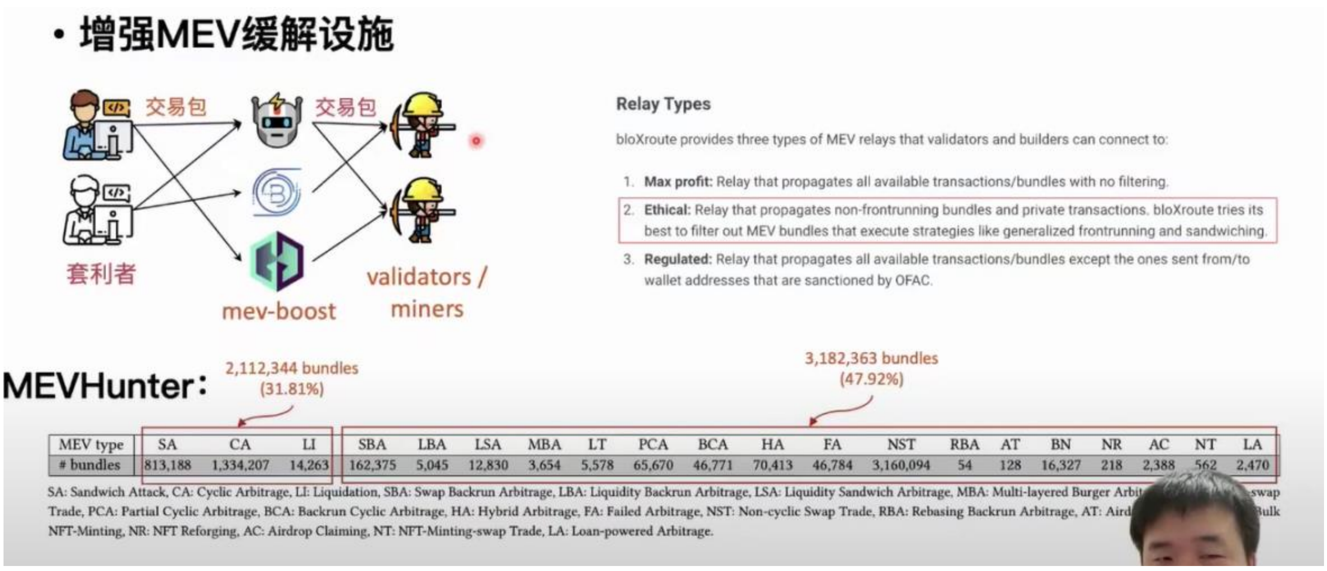
Under consistent configurations, our tools could achieve near 100% precision and completeness. However, while other tools like Etherscan could achieve satisfactory precision close to 100%, they missed out on many DeFi actions. Etherscan isn't open source, so we speculate it might use manual analysis to summarize rules for identifying DeFi actions, inevitably missing types not covered by human reviewers. It's worth noting that Etherscan doesn't provide an automated interface, so large-scale recognition tasks can't be completed directly. DeFiRanger, which relies solely on heuristic rules for recognition, doesn't satisfy in terms of precision and completeness. After experimenting with ActCluster, we found that through four iterative analyses, only 2,000 transaction packages needed to be analyzed to identify 17 unknown MEV activities. When ablating some of its modules, we might need to manually analyze up to 170,000 transaction packages to identify the previously mentioned 17 unknown MEV activities.

Empirical Analysis and Applications

What specific applications does our method of identifying unknown types of MEV activities have? Firstly, can it enhance the existing MEV mitigation measures to defend against some MEV activities?

Secondly, by utilizing the analysis results, can we more comprehensively analyze the impact of MEV activities on the blockchain ecosystem, including the impact on blockchain forks, reorganizations, and the financial security of users?

Earlier, we mentioned MEV boost, where network attackers operate tools to get transaction packages from users and then distribute them to the miners and validators connected to them. Relayers will exclude transaction packages containing MEV activities from the transaction packages they receive. Through this approach, they aim to reduce the negative impact of MEV activities on the blockchain. The main idea in this step is to design corresponding rules to detect MEV activities in the transaction packages based on existing MEV activity definitions. Clearly, these relayers cannot exclude transaction packages containing unknown MEV activities. Based on our workflow, we designed a tool called MEVHunter. This tool can detect new types of MEV activities from the transaction packages.



• 增强MEV缓解设施

Relay Types

bloXroute provides three types of MEV relays that validators and builders can connect to:

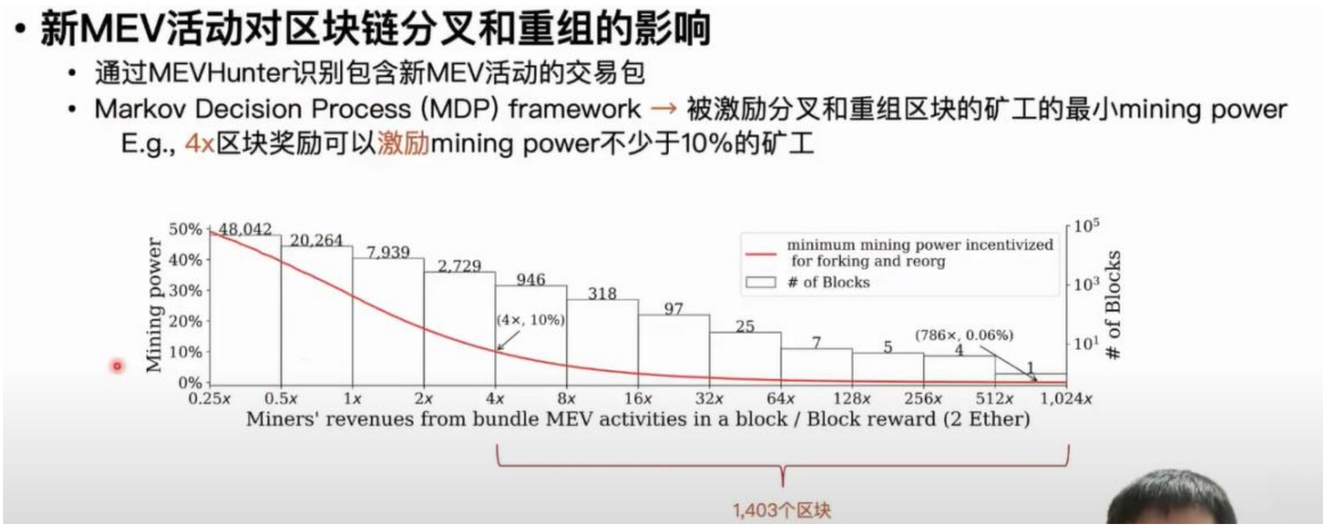
1. **Max profit:** Relay that propagates all available transactions/bundles with no filtering.
2. **Ethical:** Relay that propagates non-frontrunning bundles and private transactions. bloXroute tries its best to filter out MEV bundles that execute strategies like generalized frontrunning and sandwiching.
3. **Regulated:** Relay that propagates all available transactions/bundles except the ones sent from/to wallet addresses that are sanctioned by OFAC.

MEVHunter:

MEV type	SA	CA	LI	SBA	LBA	LSA	MBA	LT	PCA	BCA	HA	FA	NST	RBA	AT	BN	NR	AC	NT	LA
# bundles	813,188	1,334,207	14,263	162,375	5,045	12,830	3,654	5,578	65,670	46,771	70,413	46,784	3,160,094	54	128	16,327	218	2,388	562	2,470

SA: Sandwich Attack, CA: Cyclic Arbitrage, LI: Liquidation, SBA: Swap Backrun Arbitrage, LBA: Liquidity Backrun Arbitrage, LSA: Liquidity Sandwich Arbitrage, MBA: Multi-layered Burger Arbi [illegible] swap Trade, PCA: Partial Cyclic Arbitrage, BCA: Backrun Cyclic Arbitrage, HA: Hybrid Arbitrage, FA: Failed Arbitrage, NST: Non-cyclic Swap Trade, RBA: Rebasing Backrun Arbitrage, AT: Aird [illegible] Bulk NFT-Minting, NR: NFT Reforging, AC: Airdrop Claiming, NT: NFT-Minting-swap Trade, LA: Loan-powered Arbitrage.

Based on the detection results, over 1 million transaction packages contain Reverse Arbitrage MEV activities. Additionally, of the 6 million transaction packages, 30% contain three known MEV activities. For the newly discovered MEV activities, nearly half of the transaction packages contain only these new MEV activities. If relayers use the MEVHunter tool, they can filter out 3 million transaction packages containing MEV activities, which can then be excluded to reduce the negative impact of MEV activities on the blockchain.

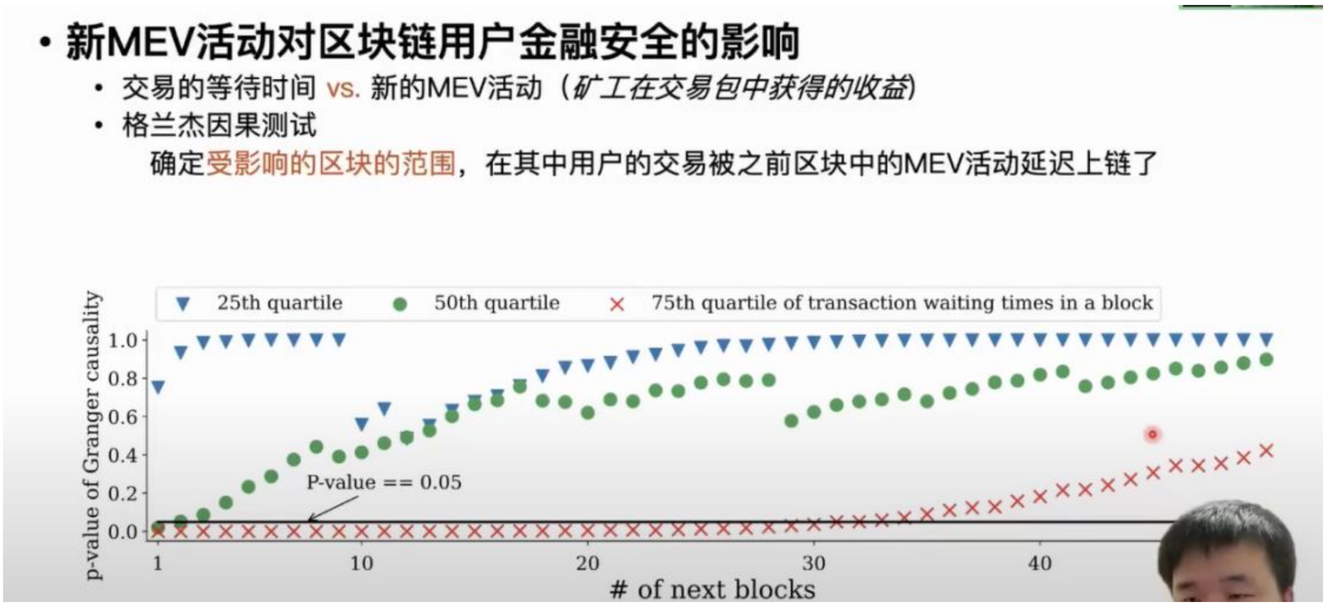


- 新MEV活动对区块链分叉和重组的影响
  - 通过MEVHunter识别包含新MEV活动的交易包
  - Markov Decision Process (MDP) framework → 被激励分叉和重组区块的矿工的最小mining power
    E.g., 4x区块奖励可以激励mining power不少于10%的矿工

Another application explores the impact of new MEV activities on blockchain forks and reorganizations. Previous studies have reported that some financially motivated miners are incentivized by the profits of MEV activities to fork and reorganize the current blockchain to carry out MEV activities and enjoy the benefits. For instance, when the MEV activity profit of a block is four times the block reward, no less than 10% of the miners will fork and reorganize this block.

First, using the previously mentioned MEV Hunter tool, we identified which transaction packages contain new MEV activities. Then, based on the profits of the miners in these transaction packages, we estimated the intensity of these MEV activities. One concept to note here is that in the transaction package mechanism, arbitrageurs usually share a portion of the MEV activity profits with miners to ensure their arbitrage transaction packages are added to the blockchain. Miners will ultimately select the transaction package with the highest profit. Using this profit, we can uniformly estimate the MEV activity in each transaction package. According to our statistics, there are over 900 blocks where the MEV profit is between four to eight times the block reward. One block even had an MEV reward over 700 times the block reward. Using a Markov decision framework, we determined that given an MEV profit, at least how many miners would be incentivized to fork and reorganize the block. We found that over 1,000 blocks could incentivize at least 10% of miners to perform block forks and reorganizations. In the most severe case, no less than 0.06% of miners would reorganize the block.

The third application investigates the impact of MEV activities on the financial security of blockchain users. MEV activities can cause delays in transaction confirmation times in the transaction pool or P2P network, posing a significant threat to users' financial security. If a user's transaction is delayed, arbitrageurs have more time to design more complex and profitable MEV activities. We first collected the waiting times of transactions. Mainly, by deploying nodes on the network, recording the first time the transaction was found on the network, and the time it was finally added to the blockchain, we calculated the waiting time. Using the waiting times of all transactions in each block's three quartiles, we organized the waiting times into a time series for each block. For each block's MEV activity, we also depicted it using the profit miners obtained from transaction packages containing new MEV activities. Through Granger causality tests, we evaluated the impact of MEV activities on transaction times. The causality test can determine whether fluctuations in one time series cause fluctuations in another and the range of its influence. We investigated whether fluctuations in MEV activities caused longer transaction wait times and the duration of this impact in subsequent blocks.



- 新MEV活动对区块链用户金融安全的影响
  - 交易的等待时间 vs. 新的MEV活动（*矿工在交易包中获得的收益*）
  - 格兰杰因果测试

    确定受影响的区块的范围，在其中用户的交易被之前区块中的MEV活动延迟上链了

The Granger causality test, when the **P** value is less than or equal to 0.05, implies that the transaction waiting time within that block was influenced and extended by previous MEV activities. Based on our analysis, when an MEV activity takes place, the waiting time for 50% of the transactions in the next two blocks will be prolonged. After the occurrence of an MEV activity, 25% of the transaction waiting times will be extended for up to the next 30 blocks. Miners or validators often place transactions with lower gas costs at the tail end of the block being packaged. The lower the gas price of a user's transaction, the more significant the impact from MEV activities, resulting in longer wait times.

In summary, we initially discussed our approach to identifying unknown MEV activities through our workflow and delved into the detailed design of the two primary modules in this workflow. We then demonstrated the efficacy of our workflow through empirical analysis and highlighted three potential applications. Presently, our workflow has led to the discovery of 17 new MEV activities.