A cat-and-mouse game

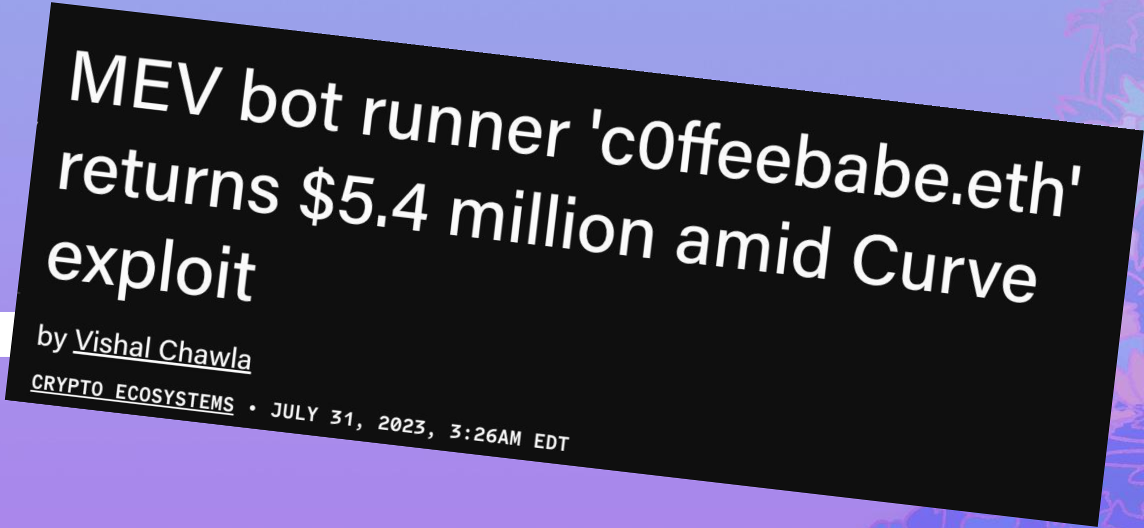
how to frontrun a transaction in the future?

Qi Su

Security Engineer, FuzzLand

Frontrunners had rescued millions of $

Frontrunners ha



MEV bot runner 'c0ffeebabe.eth' returns $5.4 million amid Curve exploit

by Vishal Chawla

CRYPTO ECOSYSTEMS • JULY 31, 2023, 3:26AM EDT

MEV bot runn 'c0ffeebabe.eth'
retu
on amid Curve

BlockSec prevents $5 million from being stolen on Paraspace

by Vishal Chawla

CRYPTO ECOSYSTEMS • MARCH 17, 2023, 6:06AM EDT

MEV bot runner 'c0ffeebabe.eth'

on amid Curve

BlockSec

from be

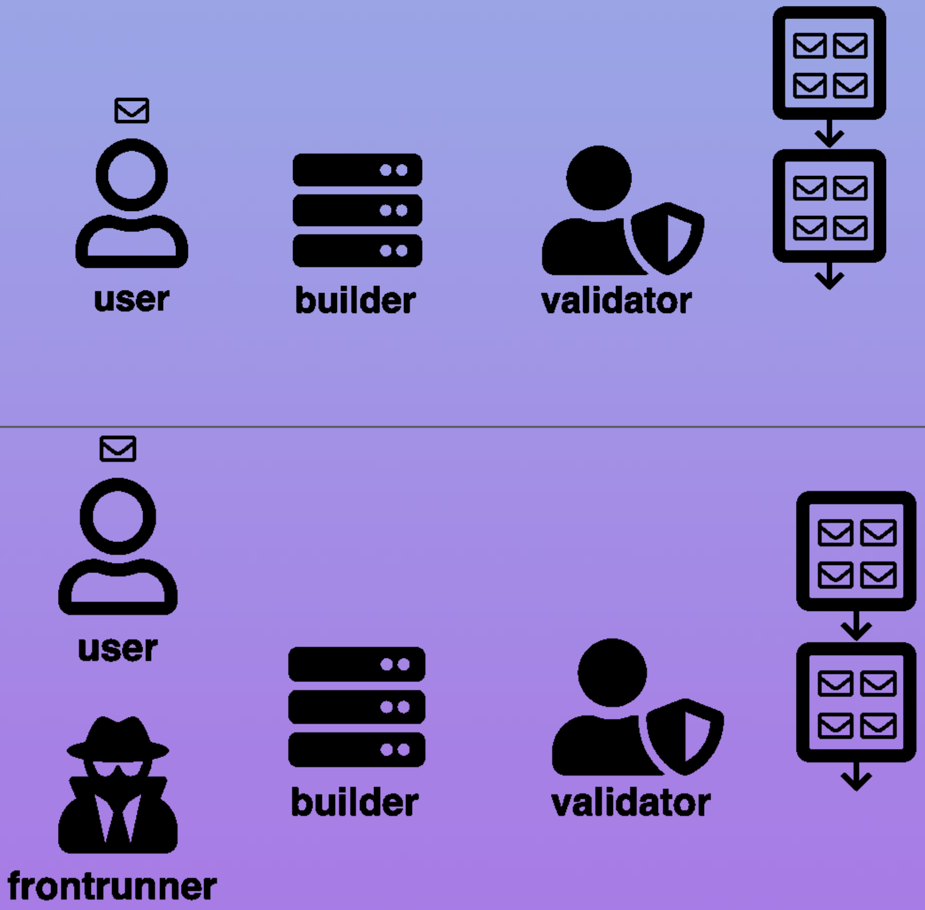
by Vishal Chawla

CRYPTO ECOSYSTEMS • MARCH 17, 2023,

A hacking incident resulted in a loss of $47 million from Kyber's concentrated liquidity pools last week.

A portion of funds' recovery was completed yesterday through negotiations with the operators of front-running bots, which extracted about $5.7 million in crypto from KyberSwap pools on the Polygon and Avalanche networks during the hack, the team said.

Background of MEV and frontrunning



How frontrunning works?

- A profitable tx is sent the mempool
- Nodes propagates this tx
- The frontrunner sees this tx
- The frontrunner copies the tx
    - Replace the beneficiary
    - Add gas price

“We will keep txs private”
—Private mempool

So that:

- Arbitrages are fair
    - Better swap paths win
- User swaps are SAFU
    - No 🥪
- Hacker’s transactions are protected
    - 😈Can’t be frontran

Is frontrunning dead?

Not on a block level!

2-phase-style attack

- Deployment (and preparation)
  - Deploy assistant contract
  - Does preparations (e.g. buy token)
- Trigger
  - Trigger a vulnerability

```
contract ExampleExp {
    function prepare() external {
        token.buy();
    }

    function trigger() external {
        victim.vuln_func();
    }
}
```

Exploit a 2-phase-style attack

- Extract all functions of a contract
    - Function signatures
- Call every function (to find trigger)

```
contract ExampleExp {
    function prepare() external {
        token.buy();
    }

    function trigger() external {
        victim.vuln_func();
    }
}
```

MEV bots vs. Hackers
🐱 vs. 🐭

- Address verification

```
function trigger() external {
    require(tx.origin == fixed_address, “no”);
    require(msg.sender == fixed_address, “no”);
    // do hacking
}
```

MEV bots vs. Hackers

🐱 vs. 🐭

- Address verification
  - Address hints

```
function trigger() external {
    require(tx.origin == fixed_address, “no”);
    require(msg.sender == fixed_address, “no”);
    // do hacking
}
```

MEV bots vs. Hackers

🐱 vs. 🐭

- Address verification
  - Address hints
- Authentication

```
function trigger() external {
    require(keccak256(abi.encodePacked(msg.sender)) == fixed_hash, “no”);
    // do hacking
}
```

MEV bots vs. Hackers
🐱 vs. 🐭

- Address verification
  - Address hints
- Authentication
  - Path inversion

```
function trigger() external {
    require(keccak256(abi.encodePacked(msg.sender)) == fixed_hash, “no”);
    // do hacking
}
```

MEV bots vs. Hackers

🐱 vs. 🐭

- Address verification
  - Address hints
- Authentication
  - Path inversion
- Parameter hiding

```
function trigger(uint256 a) external {
    victim.vuln_func(a /*, … fixed values*/);
}
```

Goal:
To find an input
to trigger a profitable path
in a contract

Workflow of fuzzing

- Generate a “random” input
- Execute the program
- Observe/analyze the execution
- Collect interesting information
- Exit if certain conditions are met; Otherwise repeat

Fuzzing =
∞ monkeys +
∞ typewriters +
∞ time

Different fuzzing purposes

Web2 (exe)	Web3 (audit)	Web3 (MEV)
Corrupt Memory	Break Invariants	Find a Profitable Path

Input generation

- Random values
- Constants
  - address(0)
  - type(uint.*).max
  - …
- Other heuristics
  - Multiples of 1e18
  - Values collected during execution
  - …

Pros and Cons

- Fast
  - Symbolic execution is slow
- Accurate
  - Concrete execution
- Easy to build a prototype

- # inputs parameters
  - Complexity increases exponentially
- Effectiveness depends on input generation
  - Poor inputs == Poor effectiveness
- Thoroughness
  - Can’t reach 100% coverage

Bring more methodologies into Web3

Thank you!

Qi Su

Security Engineer, FuzzLand

@publicqi