TUM BLOCKCHAIN CONFERENCE 25

EXPLORING MEV ON ETHEREUM:

ANALYSIS OF SANDWICH ATTACKS

Problem Statement

Expectation

- [ ] Get data
- [ ] What is MEV
- [ ] Analyze data
- [ ] Analyze results
- [ ] Conclusion

Reality

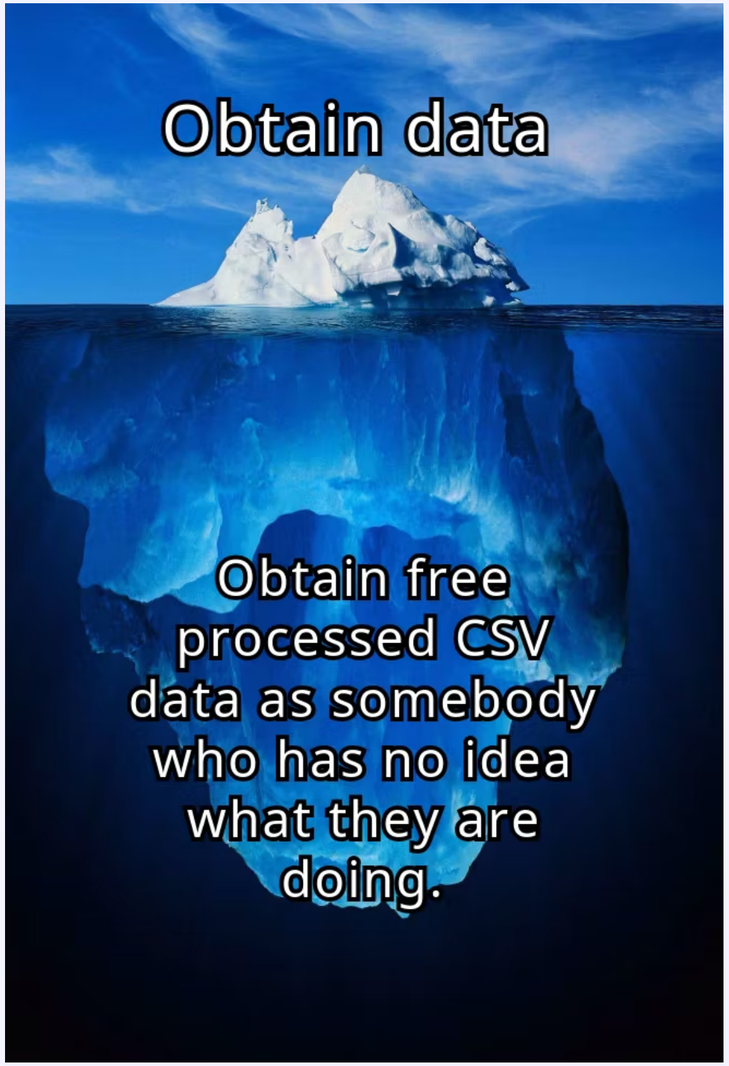
- [ ] get data 1
- [ ] what data ?
- [ ] get data 2
- [ ] get free data 3
- [ ] what is MEV
- [ ] video what is MEV
- [ ] get totally different data 4
- [ ] analyze data
- [ ] get more data
- [ ] some nice plots
- [ ] find sandwich
- [ ] better understand data
- [ ] do some heuristics
- [ ] plotting
- [ ] write it down
- [ ] correct mistakes in code
- [ ] look up things at ETH scan
- [ ] Conclusion

SEARCH

Obtaining Data

BigQuery

Blockchain ETL



Analysis of the data

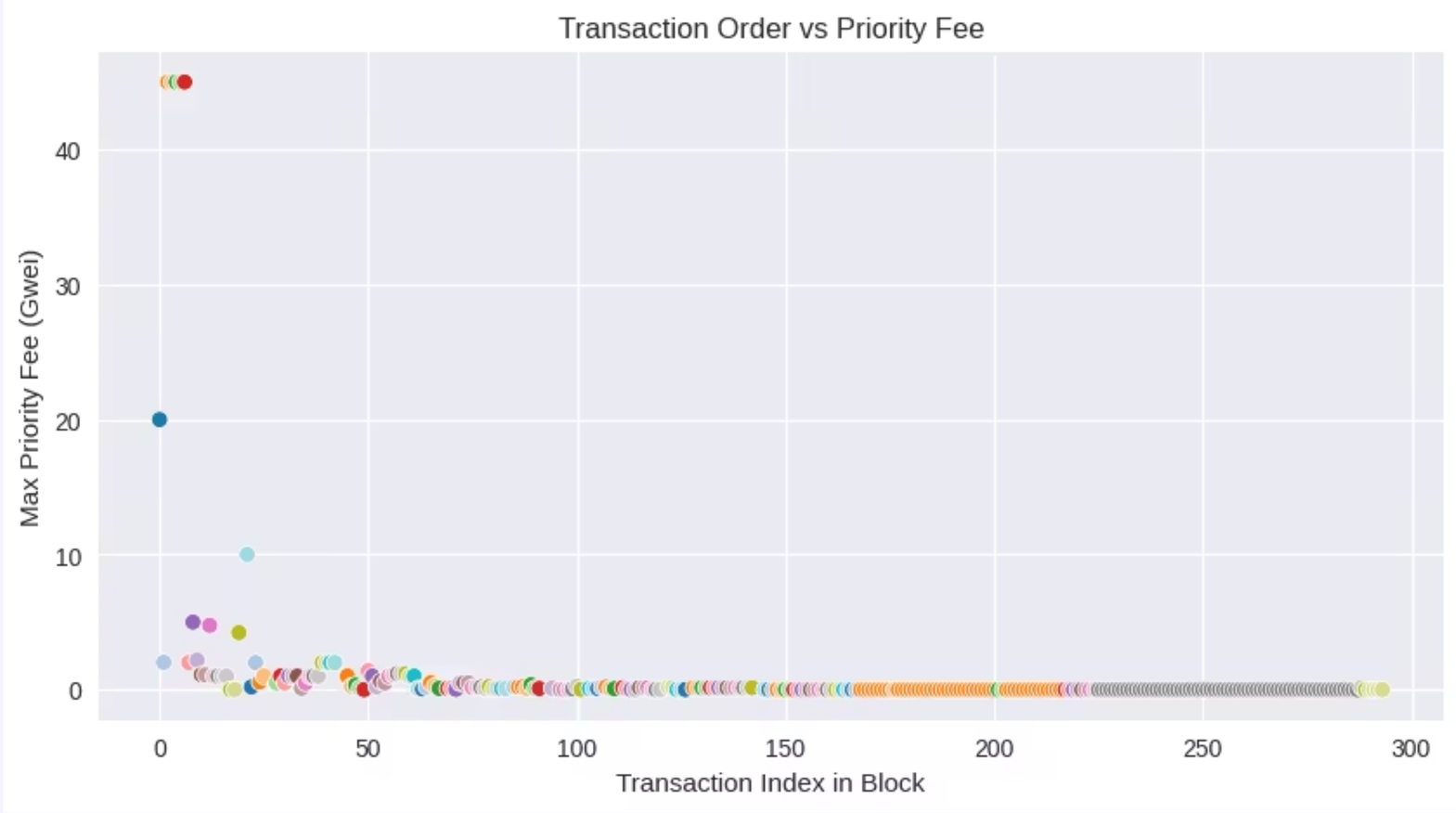
Query results Blockchain metrics

total_blocks
23340825

104,000 transactions

across blocks

23300000–23300500



Sandwich Heuristics

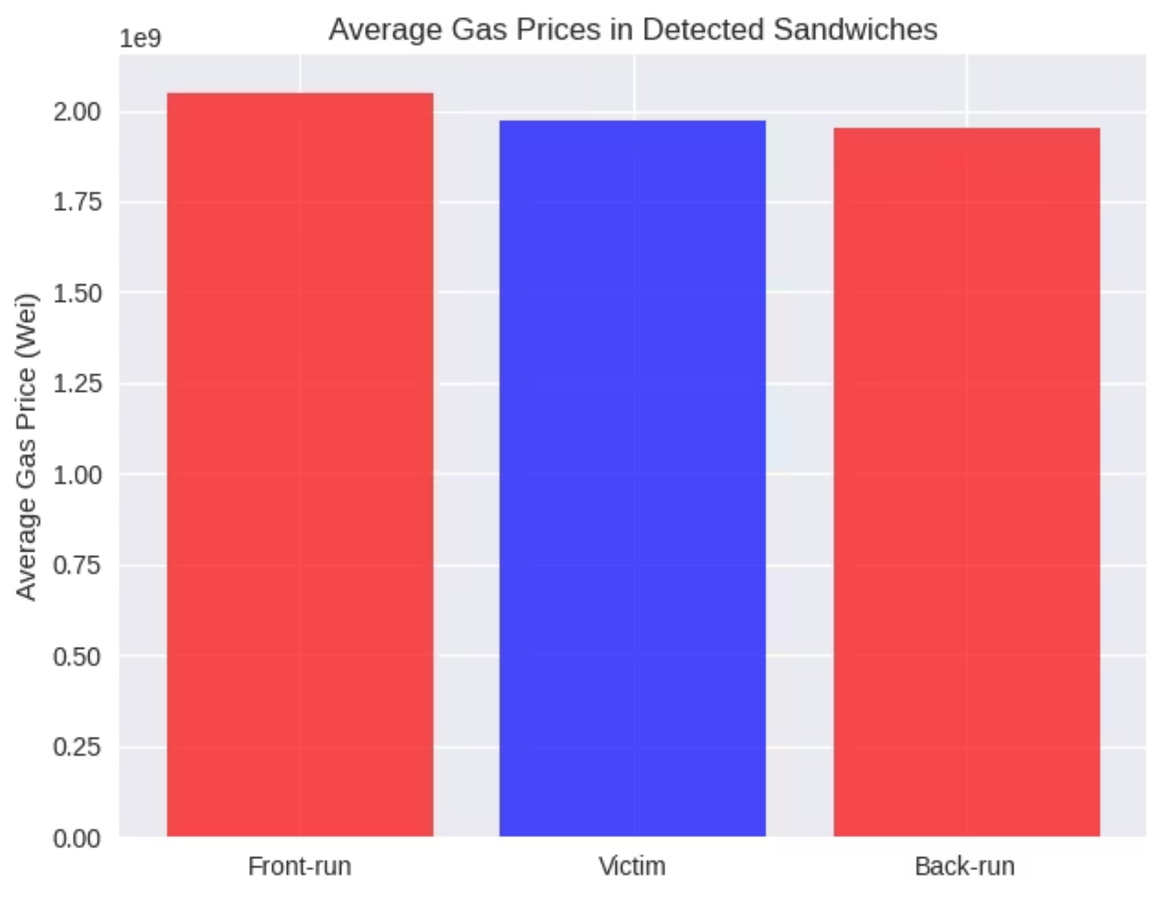
- **transactions with non-zero input and gas tip**, and valid recipient addresses.
- Analyze **three consecutive transactions** within the same block:
  1. **Front-run (attacker)**
  2. **Victim transaction**
  3. **Back-run (attacker)**

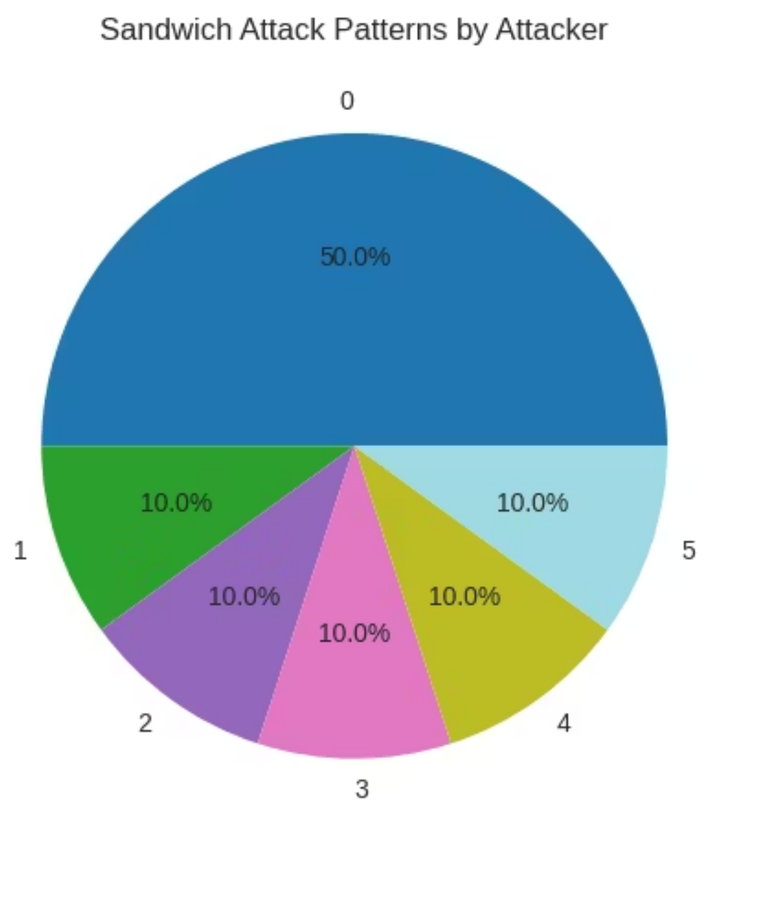
- **Front-run criteria:**
  - Gas tip higher than victim's and in the **top 10% of the block**.
  - Must occur **before victim** in transaction order.
- **Back-run criteria:**
  - Sent by s**ame attacker as front-run or directed to them.**
  - **Gas tip close (~10% tolerance) to victim's gas tip.**
  - **M**ust occur **after victim** in transaction order.

- **Victim criteria:**
  - Transaction value above minimum threshold.
  - Different sender from attacker.

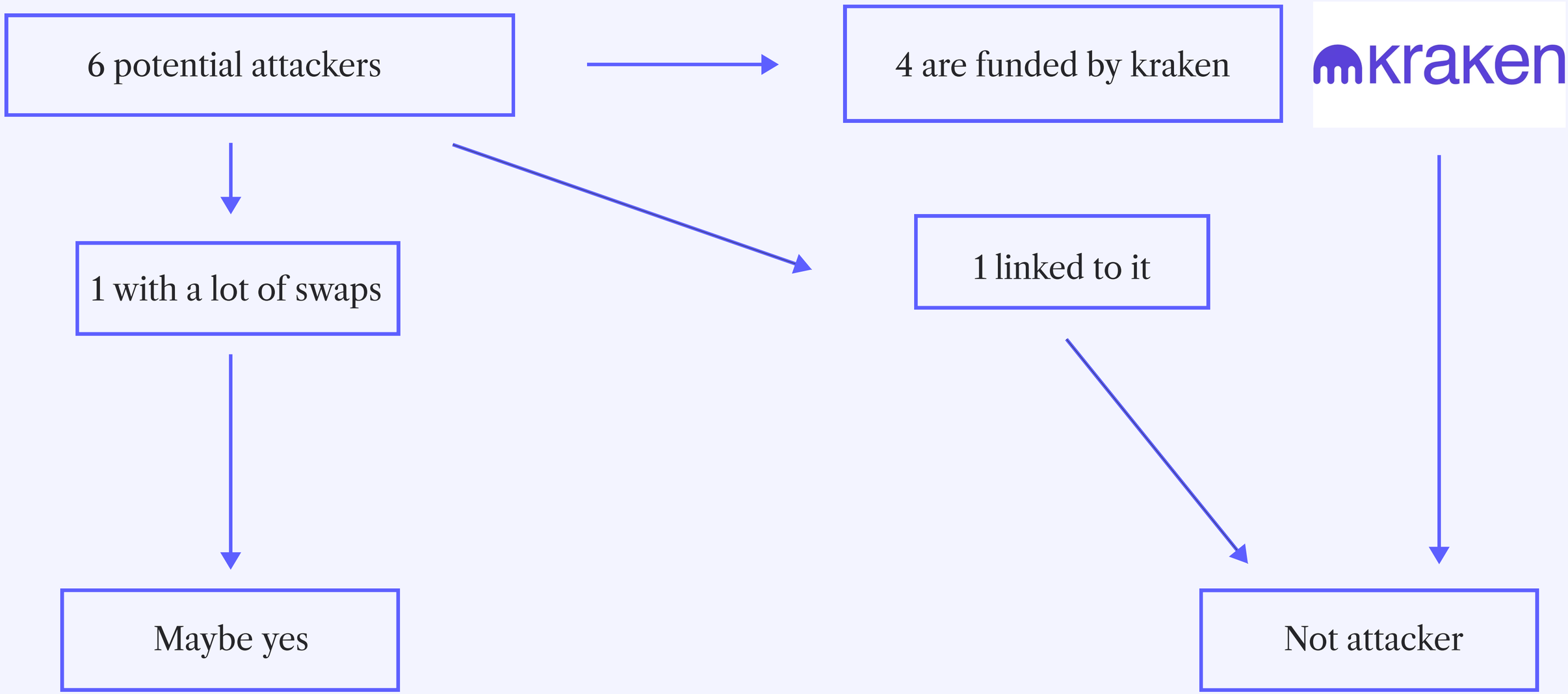
```python
df_filtered = df[
    (df["input"] != "0x")
    & (df["to_address"].notnull())
    & (df["max_priority_fee_per_gas"].notnull())
    & (df["max_priority_fee_per_gas"] > 0)
]

def find_sandwiches_strict(block_df, min_victim_value=0):
    sandwiches = []
    n = len(block_df)

    if n < 3:
        return sandwiches

    # Compute 90th percentile of gas prices in the block
    fee_90pct = block_df["max_priority_fee_per_gas"].quantile(0.9)

    for i in range(n - 2):  # only check 3 consecutive transactions
        front = block_df.iloc[i]
        victim = block_df.iloc[i + 1]
        back = block_df.iloc[i + 2]

        if (
            victim["value"] < min_victim_value
            or front["from_address"] == victim["from_address"]
        ):
            continue

        if front["max_priority_fee_per_gas"] < victim["max_priority_fee_per_gas"]:
            continue
        if back["max_priority_fee_per_gas"] > victim["max_priority_fee_per_gas"]:
            continue
        if (
            front["max_priority_fee_per_gas"] < fee_90pct
        ):
            continue

        back_matches_attacker = (
            back["from_address"] == front["from_address"]
            or back["to_address"] == front["from_address"]
        )
        back_gas_ok = (
            abs(back["max_priority_fee_per_gas"] - victim["max_priority_fee_per_gas"])
            / victim["max_priority_fee_per_gas"]
            <= 0.1
```

Results





Results



Results

Step	Wallet	Action	max priority fee (Gwei)	Index	Description
Front-run	`0×65A8F07B...`	Swaps 0.11592 ETH for 58,793 CHILLHOUSE on Uniswap V2	3.000	3	Attacker buys tokens before victim
Victim	`0xbabe01c4...`	Swaps 4,445.8 CHILLHOUSE for 0.00885 ETH	2.202	4	Victim executes trade
Back-run	`0×65A8F07B...`	Transfers 58,793 CHILLHOUSE	2.000	5	Attacker sells tokens after victim

CONCLUSION

Before

I heard blockchain, ETH, Bitcoin

After

MEV

Arbitrage

Front-Running

Gas Fees

Sandwich Attack

Back-Running

Mempool

Transactions

Eth Scan

And more ...

THE END

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It’s free)