Principles of Distributed Ledgers

Lecture 5: Technical and Economic Security

Lewis Gudgeon, **Daniel Perez**, Paul Pritz, **Sam Werner**

February 24, 2023

Imperial College London

DeFi Hacks

Global regulators target blockchain-based 'decentralised finance'

Anyone Seen Tether's Billions?

Regulatory risks grow for DeFi as a 'money laundering haven'

DeFi Protocol Compound Mistakenly Gives $162 Million To Users, CEO Begs Them To Give It Back

CREAM Finance Exploited for $130M in DeFi's Third-Largest Hack

Legislation on stablecoins needed 'urgently', say top US regulators

Defi Protocol Harvest Finance Hacked for $24 Million, Attacker Returns $2.5 Million

WANTED! $1m bounty on offer for information on cryptocurrency firm tether's so-called 'stablecoin' backing

$600 million gone: The biggest crypto theft in history

China 'Banned' Crypto. Can The SEC Try Doing The Same?

DeFi Protocol Pickle Finance Hacked For $20 Million

DeFi Pessimist

Binance Chain DeFi Exchange Uranium Finance Loses $50M in Exploit

90% of NFTs Will Be Worthless in 3 to 5 Years, Coinbase Cofounder Warns

Different types of exploits

- Technical exploits
  - Smart contract vulnerabilities
  - Single transaction exploits
  - Ordering exploits
- Economical exploits

Technical security

Definition

Informal Definition

Technical security = secure from an attacker who is limited to atomic actions (e.g., not possible to steal assets)

- Technical security is about whether an on-chain system can be exploited within a **single tx** or a **bundle of txs** in a block
- Technical attacks are risk-free b/c outcomes are binary for attacker
  - Either attack is successful = profit $$
  - Or it reverts = only pay gas fee

Smart contract vulnerabilities

Logical bugs

There are many types of logical bugs that have led to exploits

- Authorization issues
- Lack of check for contract invariants
- Lack of proper success/failure checks
- Integer manipulation issues (rarer since Solidity 0.8)

Parity wallet “hack”

Over 513,774 ETH (~$10M) got stuck because of an authorization issue in the Parity wallet library

```
// this function had not been called
function initWallet(address[] _owners, uint _required,
                    uint _daylimit) only_uninitialized {
  initDaylimit(_daylimit);
  initMultiowned(_owners, _required);
}
function kill(address _to)
        onlymanyowners(sha3(msg.data)) external {
  suicide(_to);
}
```

Compound bug

Compound distributed ~90M USD worth of extra COMP rewards

```
if (supplierIndex == 0 && supplyIndex > compInitialIndex) {
    supplierIndex = compInitialIndex;
}
Double memory deltaIndex = Double({
        mantissa: sub_(supplyIndex, supplierIndex)});
uint supplierTokens = CToken(cToken).balanceOf(supplier);
uint supplierDelta = mul_(supplierTokens, deltaIndex);
```

Compound bug

Leshner
@rleshner

If you received a large, incorrect amount of COMP from the Compound protocol error:

Please return it to the Compound Timelock (0x6d903f6003cca6255D85CcA4D3B5E5146dC33925). Keep 10% as a white-hat.

Otherwise, it's being reported as income to the IRS, and most of you are doxxed.

1:13 AM · Oct 1, 2021

269 Retweets **1,227** Quote Tweets **1,880** Likes

Reentrancy

- Vulnerable contract calls an external function or sends ETH to another address
- Attacker contract's function is called
- Attacker contract makes re-entrant call to attacker
- Vulnerable contract reads state that is now stale (not properly updated)

Example: Reentrancy

Vulnerable contract

```
function withdrawAll() {
  uint256 amt = balances[msg.sender];
  msg.sender.call{value: amt}("");
  balances[msg.sender] = 0;
}
```

Attacker contract

```
uint256 count = 0;
function attack() {
  vulnerable.withdrawAll();
}

receive() payable {
  count += 1;
  if (count < 10) {
    vulnerable.withdrawAll();
  }
}
```

Cream Finance exploit

Cream Finance (lending protocol) got exploited 20M USD with a re-entrancy attack

Call trace (source: Peckshield)

Defenses

- Testing
  - Unit/integration testing
  - Fuzz testing
  - Formal verification
- Security audits
- Bug bounties

Transaction Ordering

Ordering

- Attacks may involve front- and/or back-running within a single block, thereby undermining the technical security of DeFi protocols
- Recall that order of transactions in a block is determined by the miner/validator!
- Many profitable opportunities exist because of the order of txs
- **Front running**: Taking profitable actions based on (typically non-public) information on upcoming trades in a market
- **Miner/Maximal Extractable Value (MEV)**: The value a miner/validator can extract from deciding tx order and inclusion

Example: Front running

1. Alice submits a transaction to perform some arbitrage across two decentralised exchanges to the mempool.
2. Bob sees Alice's pending transaction and submits a copy of same transaction with a higher priority fee.
3. Bob's transaction is included before Alice's transaction and he profits from the arbitrage opportunity.

Example: MEV

1. Alice submits a transaction to perform some arbitrage across two decentralised exchanges to the mempool.
2. The miner/validator sees Alice's pending transaction and includes a copy of same transaction (created by the validator).
3. The validator's transaction is included before Alice's transaction and he profits from the arbitrage opportunity.

Note: The validator could also decide to not include Alice's transaction at all.

Sources of MEV

DeFi applications give rise to many new sources of MEV, such as:

- Decentralised exchanges present atomic arbitrage opportunities
- Liquidation mechanisms (e.g., in stablecoins, PLFs) = arbitrage opportunities

Extracting MEV

Miners/Validators can:

- Exclude transactions
- Determine order of txs
- Insert their own txs

Anyone can:

- Front run txs
- Back run txs

Single Transaction Attacks

Single Transaction Attacks

- The attack logic can be **executed in a single tx**
- Success of attack does not depend on order of transactions in block (or on any other transaction in the block)
- Examples:
  - AMM manipulation attacks
  - Governance attacks

Single transaction AMM manipulation

- An attacker may temporarily imbalance an AMM to modify the price of an asset (e.g. using a flash loan)
- Should the AMM be used by some protocol as a price oracle an attack vector may exist
- Attacker can interact with the protocol at a manipulated price and potentially make a profit
- Attacker restores the balance of the AMM at the end of the transaction

Example: Harvest Finance Exploit

Example: Attacker

Details: rekt.news

Governance attack

- DeFi protocols often issue tokens that are used for governance (i.e. required for creating proposals and voting)
- An attacker may obtain an amount of governance tokens sufficient to propose and execute malicious contract code and potentially steal a contract's funds
- Governance tokens could be borrowed from PLFs or sourced from AMMs via flash loans

Example: Build Finance DAO

Transaction Ordering Attacks

Transaction Ordering Attacks

Success of attack **depends on the order** of transaction(s).

Types of transaction ordering attacks:

- Displacement attacks
- Sandwich attacks

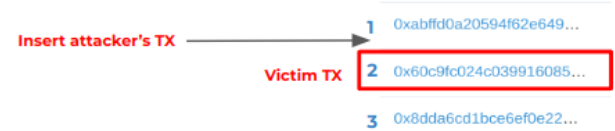
Displacement attack

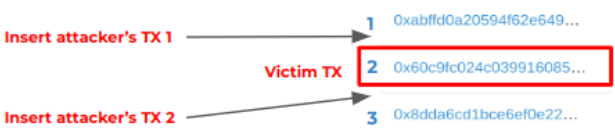
An attacker front runs some target transaction where the success of the attack does not depend on whether the target transaction is executed afterwards or not.

Examples:

- Trying to register a domain name
- Trying to claim a bug bounty

Front running and Back running





Sandwich attack

An attacker alters the deterministic price on an AMM **prior to and after some other target transaction has been executed** in order to profit from temporary imbalances in the AMM's liquidity reserves

Example: Sandwich attack

1. Alice submits a transaction to swap asset X for asset Y.
2. Bob sees the pending transaction of Alice and submits one transaction that needs to be executed before (with a higher priority fee) to purchase a large amount of asset Y.
3. Bob's transaction gets executed first, driving up the price of asset Y. Alice's transaction then gets executed and she buys asset Y at a higher price, driving up the price.
4. Bob submits a transaction in which he then sells asset Y at a higher price, making a profit.

Example: Sandwich attack

Swapping WETH for FARM tokens on Uniswap:

- TX1 (*Open sandwich*): Attacker swaps 220 WETH for 1234 FARM.
- TX2 (*Target transaction*): Victim swaps 153 WETH for 782 FARM (price: 1 WETH = 5.11 WETH)
- TX3 (*Close sandwich*): Attacker swaps 1234 FARM tokens for 234 WETH (price: 1 WETH = 5.27 FARM)

Issues with extractable value from tx ordering

The profit that can be extracted from tx ordering can result in several issues:

- Bidding wars for which tx(s) get included result in gas price volatility
- Failed txs still get included and take up block space unnecessarily (they revert)

Defenses

- A private mempool and communication channels can be used
- Example: Flashbots

Economic security

Definition

Informal Definition

Economic security = not profitable for an attacker who can perform non-atomic actions to manipulate the protocol into unintended states

- Economic security is about an exploiting agent who tries to manipulate the incentive structure of the protocol to profit (e.g., by stealing assets)
- Economic exploits are non-atomic
- They have upfront tangible costs and are **not risk-free**
  1. The attack may fail depending what else happens in the time period
  2. The attacker may mis-estimate the market response

Technical vs Economic – what's different?

Technical exploit: attacker finds sequence of contract calls that leads to a profit

- Single tx or bundle of txs

Formal model of contracts is "enough" – can be hard CS problems to work out optimal attack

Technical vs Economic – what's different?

Economic exploit:

- Attacker performs multiple actions at different 'times'
- But doesn't control what happens between the actions
- No guarantee final action is profitable (i.e. not risk free)

Need models of markets, which we can't model exactly vs. formally verifying contract code

Technical vs Economic – a simple example

A technical exploit: a protocol uses the instantaneous AMM price as an oracle, and an attacker performs a (atomic) sandwich attack to steal assets

An economic exploit: a protocol uses a time-weighted average AMM price as an oracle. An attacker manipulates this price over time and may be able to steal assets