ethereum

vienna

Dao Attack and Consequences

Agenda

General Introduction

Updates

Dao Attack and Consequences

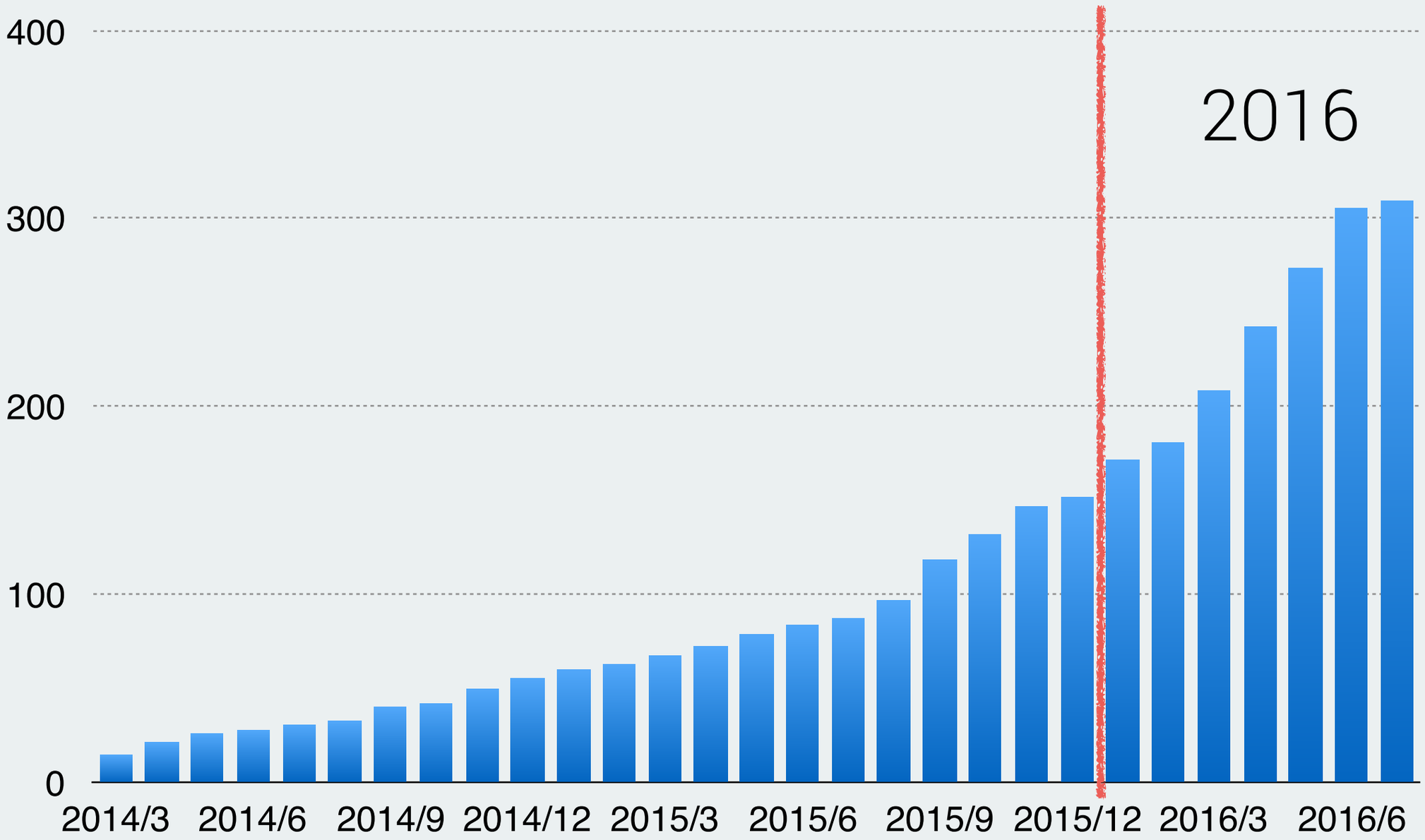
Socialising

ethereum

vienna

Updates

300


400
300
200
100
0
2016
2014/3
2014/6
2014/9
2014/12
2015/3
2015/6
2015/9
2015/12
2016/3
2016/6

Geth

Quite a few updates since the last meetup

Make sure you're on the latest version

Syncing should work much better

Latest: Geth 1.4.9: The Network strikes back

UI Wallet

Newest version: Wallet 0.7.6 (Beta 20)

Warns about pre-homestead wallet contracts

Used tx.origin. Now deemed unsafe.

=> Move your ether to a new wallet

Doesn't freeze on complicated contracts anymore

LES

Light Client is ready for testing

see 'Light Client Public Test' in wiki at
github zsfelfoldi/go-ethereum

Workshops

No advanced people at Workshop #2

WS3 now postponed to after the summer

All events will be on meetup soon

Date finding once there are enough RSVPs

(probably not before september / october)

Don't RSVP if you don't intend to attend

Workshops

Workshop #1: Contract Development for Beginners

- Requirements: Basic Understanding of Ethereum
- Solidity Basics

Workshop #2: From Idea to Contract (Advanced)

- Requirements: Basic Understanding of Solidity
- Mapping the real world to ethereum concepts
- Advanced Solidity

Workshop #3: From Contract to DApp

- Requirements: Basic Understanding of Solidity, HTML/JS, node.js
- Interfacing with Ethereum using web3.js
- Auxiliary Technologies: IPFS, Whisper and Swarm

New Location

Starting with the next meetup we will move to

Neubaugasse 64

Ethereum ATM there or MQ (soon)

ethereum

vienna

Dao Attack and Consequences

DAO Recap

What is the DAO?

Does not produce anything by itself

Funds proposals

Distributes revenues to its owners

What is the DAO?

Owners of the DAO - token holders (TH)

TH vote on proposals

- Voting power proportional to investment
- Necessary quorum scales with proposal size
- TH can “split” the DAO if they’re unsatisfied

Curator

Every DAO has one curator

Controls a whitelist of where funds can be sent

Should protect against various attacks

Plays a vital role in White Hat Plans

If some TH are unsatisfied with curator

submit a proposal to replace curator

special proposal, regular voting

Splitting

If the new curator proposal is accepted

the recipient in the proposal becomes the curator

otherwise the curator remains the same

but if you voted yes for the new curator

Splitting

but if you voted yes for the new curator

you can split from DAO

Your ether is sent to the new DAO with the new curator

Reward Tokens are transferred to the new DAO

Reward Tokens represent rights to reward income

Old DAO Token (**not** Reward Tokens) are burned

DAO Attack

Attack

June 17th

Attacker took ~3.6M ether from ~11.5M

Valued at almost 80M $ at the peak of the bubble (22$)

Unknown why the attacker stopped

Almost 29000 transactions involving the DarkDAO

Attack

Clever trick involving multiple functions across multiple contracts

Exploit begins in splitDAO function

```
function splitDAO(
    uint _proposalID,
    address _newCurator
) noEther onlyTokenholders returns (bool _success) {
```

Attack

… some other code

Funds for child DAO are calculated

from tokens of sender, splitBalance and totalSupply

```
// Move ether and assign new Tokens
uint fundsToBeMoved =
    (balances[msg.sender] * p.splitData[0].splitBalance) /
    p.splitData[0].totalSupply;
if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
    throw;
```

Eth Transfer to child DAO via createTokenProxy

Attack

… some other code

Afterwards withdrawRewardFor sender

THEN burn tokens

```
// Burn DAO Tokens
Transfer(msg.sender, 0, balances[msg.sender]);     This line does NOTHING
withdrawRewardFor(msg.sender);     Tokens still owned at this point
totalSupply -= balances[msg.sender];
balances[msg.sender] = 0; Here the actual burning happens
paidOut[msg.sender] = 0;
return true;
```

Attack

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])
        throw;

    uint reward =
        (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];

    reward = rewardAccount.balance < reward ? rewardAccount.balance : reward;

    paidOut[_account] += reward;
    if (!rewardAccount.payOut(_account, reward))
        throw;

    return true;
}
```

_account is the attack contract

```
function payOut(address _recipient, uint _amount) returns (bool) {
    if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
        throw;
    if (_recipient.call.value(_amount)()) {
        PayOut(_recipient, _amount);
        return true;
    } else {
        return false;
    }
}
```

.call sends all possible gas
arbitrary computation possible
_recipient is the attack contract

Attack

Attack contract has a default function

calls splitDAO every time

```
// Move ether and assign new Tokens
uint fundsToBeMoved =
    (balances[msg.sender] * p.splitData[0].splitBalance) /
    p.splitData[0].totalSupply;
if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
    throw;
```

balance still not updated

splitData constant

funds sent out again

```
// Burn DAO Tokens
Transfer(msg.sender, 0, balances[msg.sender]);
withdrawRewardFor(msg.sender);
totalSupply -= balances[msg.sender];
balances[msg.sender] = 0;
paidOut[msg.sender] = 0;
return true;
```

next reentry

only reached after attacker stops the recursion

burning of token

Attack

Before ending the recursion the attack contract transfers token to another address

Afterwards the token are sent back to the attack contract

2 attack contracts for efficiency

=> Attack can be done arbitrary number of times

```
// Burn DAO Tokens
Transfer(msg.sender, 0, balances[msg.sender]);
withdrawRewardFor(msg.sender);
totalSupply -= balances[msg.sender];
balances[msg.sender] = 0;
paidOut[msg.sender] = 0;
return true;
```

transfers token to another address

"burning of token", but it's already 0

Counterattack

June 21st

Group of white hats (from eth foundation + slock.it)

Same exploit used as attacker

7.6M ether in WhiteDAOs

but BlackHat also voted YES on split proposal

Begun the DAO Wars have

After the creation period BlackHat can drain the WhiteDAO

DarkDAO curator identified

=> WhiteHats can probably drain the DarkDAO as well

Both parties can block each other (almost) indefinitely

Every 34 days there would be twice as many DAOs

2000 DAOs after one year

2M DAOs after two years

At some point the gas cost would probably not worth the ether per DAO

Ways Forward

NoFork

SoftFork (now probably dead)

HardFork

NoFork Timeline

July 15th	First ETH leave a DAO contract
July 18th	End of WhiteHatDAO Creation Period
July 25th	Attacker can start draining the WhiteHatDAO
August 30th	First possible withdrawal of DarkDAO ETH (if not prevented by WhiteHats)

SoftFork

Collusion of a mining majority to

censor any transactions that

would lower the balance of any DAO account

Would temporarily burn all ether in DAO contracts

Potential whitelist for withdrawal

Could have retrieved the ETH from DarkDAO as well

SoftFork

Vote by the mining pools

=> Yes, at first

But then a vulnerability was discovered

SoftForks could create DOS problems

Idea probably dead now

HardFork

Change of the consensus protocol

(due an application level bug)

At a certain block height

all funds from all daos (including extraBalance)

moved back to main DAO

mainDAO code replaced by refund contract

HardFork

Distribution of ether determined off-chain at block X

Splits and Proposals executed after X ignored

Optional 2nd list with multipliers for panic splits

Becomes much more difficult after July 15th

Pro-Fork Arguments

ETH in DAO was supposed to be for Startups

Could destroy trust in ethereum

Use case of Attacker was not intent of the contract

Attacker could hurt Casper PoS

SEC and other Investigations

Anti-Fork Arguments

Code is law

Could destroy trust in ethereum

Attacker did nothing wrong

Would break Ethereum's social contract

Too many things could go wrong

DAO Token

Still trading at 0.0078

78% of initial price

=> Market still considers recovery likely

Can be traded on kraken and poloniex

github.com/ahirner/ethereum