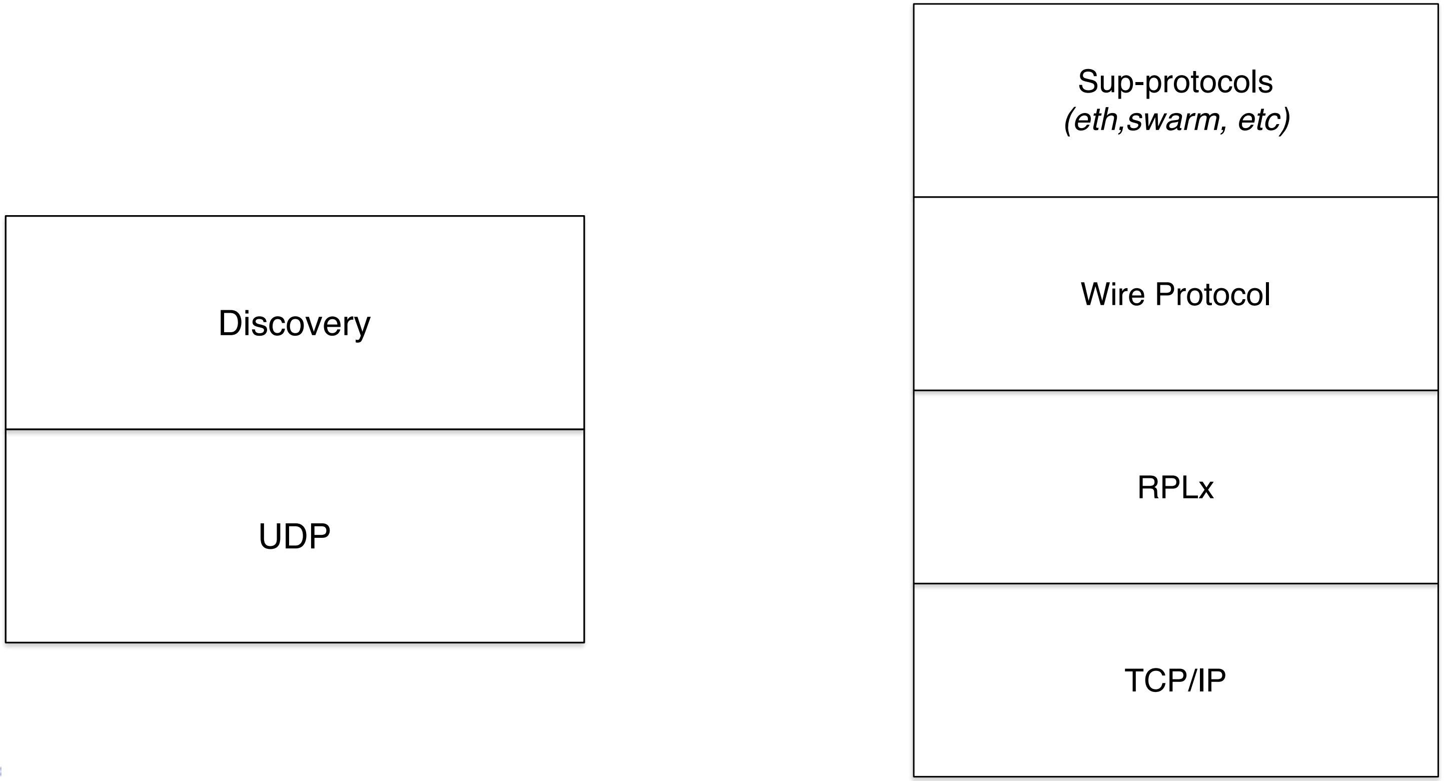
Ethereum Networking

CONSENSYS

Special Thanks

Raúl Kripalani

A tale of two stacks


Discovery
UDP
Sup-protocols
(eth,swarm, etc)
Wire Protocol
RPLx
TCP/IP

Discovery

- Kademlia based routing table
- Boot nodes

Kademlia

- p2p protocol
- node lookup and DHT (Kademlia routing table)
  - specifies the structure of the network
  - exchange of information
- UDP Communication
- https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

Kademlia Routing Table

- table divided in k-buckets, where each bucket holds nodes at a distance from us
- Distance
    - XORs Node IDs
    - (Ethereum Keccack hash of nodes)

Kademlia Messages sent

(will have responses)

- PING
- FIND_NODE
- STORE
- FIND_VALUE

Kademlia Bootstrap

- To join network
  - Connect to bootstrap node
  - Send a Find Node request of itself
  - Refresh its own buckets (lookup a random value)

How it’s implemented in Ethereum

Implemented

- Ping (with pong)
- Find Node (find neighbours)

Not Implemented

- Store
- Find Value

Discovery

- Two sets of paired messages

Ping	Pong
FindNeigbhours	Neighbours

- Messages are signed with the node private key

Bonding

- Node must have successfully responded to a Ping/Pong pair before Neighbour messages work.
- After a Ping/Pong pair, the node may get added to the routing table.
- If a node doesn't get added to a routing table it won't respond to Neighbour messages.

Message format

- Type (byte)
- Data
- EC Signature of type and data (SECP256K1)
- hash of signature, type and data

Ping

- Initiate Bonding process
- Data
    - from, to, expiry
- node may be added to the routing table when received

Pong

- Respond to ping
- Data
    - to, ping hash, expiry
- Successfully Bond on receipt of the pong if the ping hash matches

FindNeighbours

- Ask for a list of neighbours for a given node
- Data
  - Target node, expiry
- If source is not a bonded node(i.e. someone in the routing table) drop message

Neighbours

- Respond with a list of neighbours closest to the target node
- Data
  - List of peers to the target node, and expiration
- On receiving neighbours, initiate the ping/pong bonding process

Bootstrap

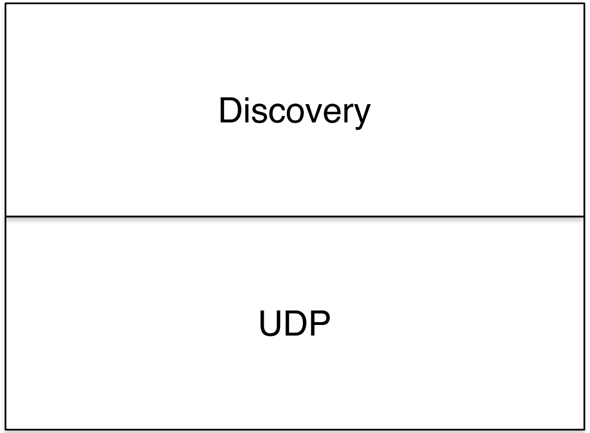
- Connect to boot nodes
- Bond to them
  - Send ping
  - On pong do a findNeighbours
- (node table is persisted to minimise bootstrap requirements)

Refresh table

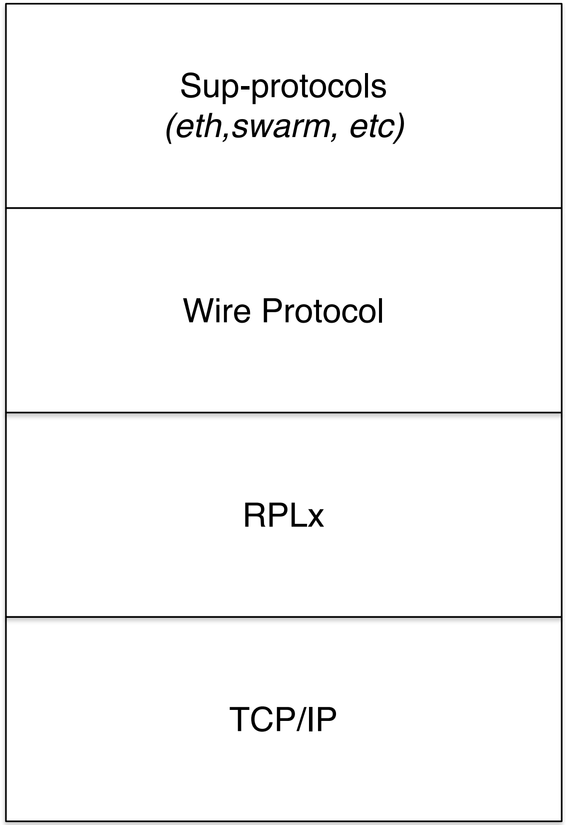
- Periodically perform the following:
- Generate a random target and query N closest nodes we know, feeding their answers into our routing table, and continue the process iteratively until we either (a) have no nodes to interrogate, or (b) we've reached a max number of iterations.

More differences

- PING PONG bonding to try to prevent DDoS
- MSB vs LSB for distance
- Reputation considers more things than how long a node is connected


Discovery
UDP

DevP2P Communications



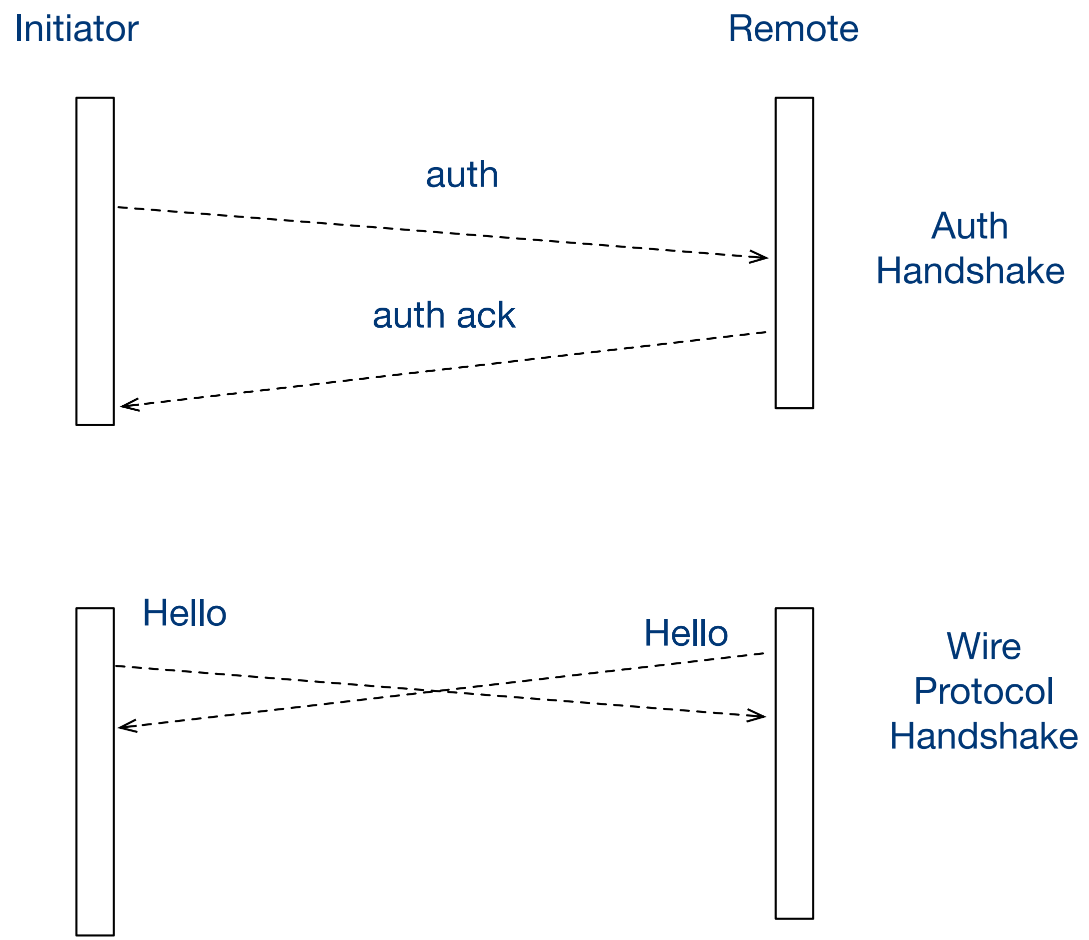
RLPx

- Encrypted Multiplexed messaging
- Initialises session
- Crypto handshake
- Messages framing
- Multiplexing of the sub protocols

Session Initialisation

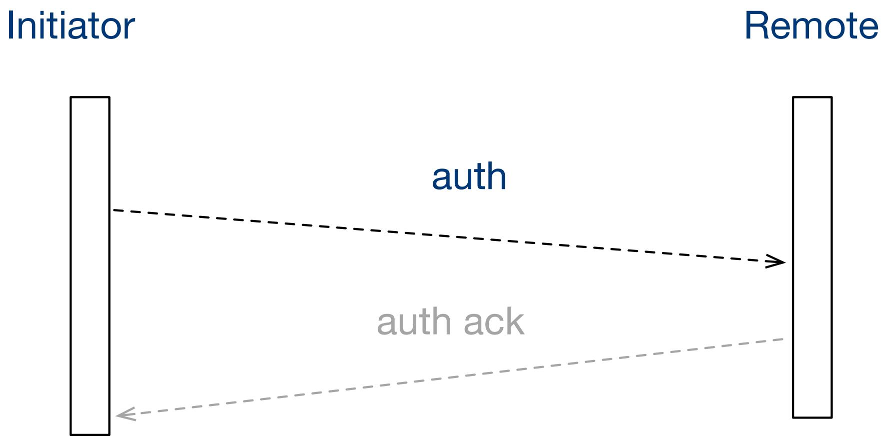
- Handshake Crypto(RLPx)
- Handshake Protocol (Wire Protocol)
- See https://github.com/ethereum/devp2p/blob/master/rlpx.md

Session Initialisation



Crypto Handshake

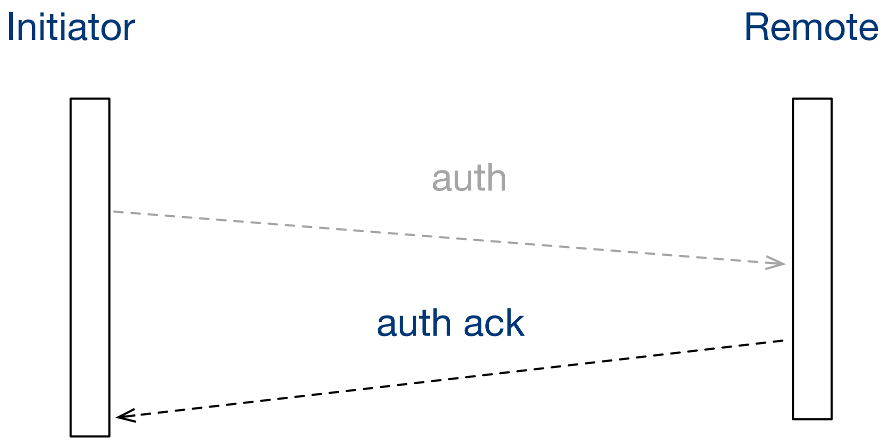
Auth



- Generates ecdhe-random, static-shared-secret, and nonce
- Send auth

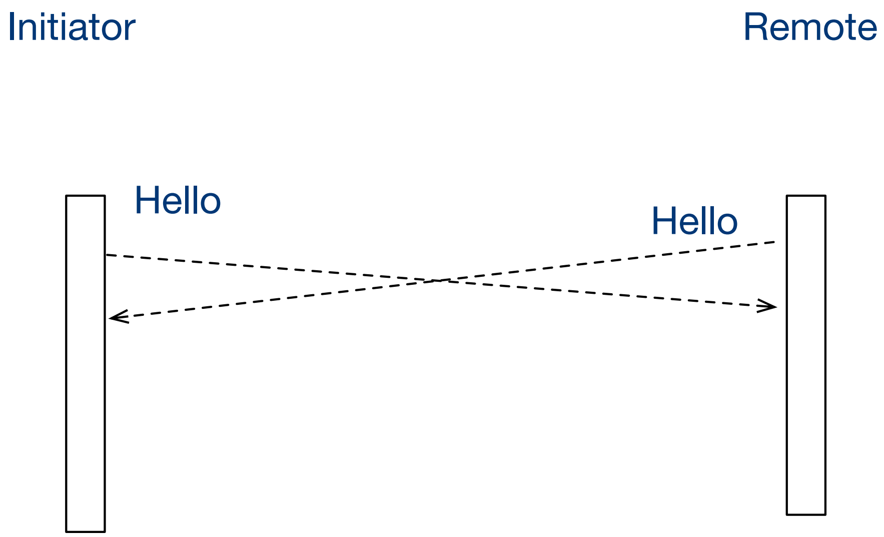
Crypto Handshake

Auth ack



- generates authAck from remote-ephemeral-pubk and nonce
- Send auth ack

Protocol Handshake



- Each node
- Derives shared-secret, aes-secret, mac-secret, ingress-mac, egress-mac
- Initiates Wire protocol and sends Hello
- Authenticates protocol-handshake (on receipt from other node)

More details

CONSENSYS

Hello Message

- Protocol Version
- Client ID
- List of capability and capability version tuples
- Port
- Node id

Sub protocol negotiation and initiation

- Collect shared capabilities between initiator and remote
- Pick highest version for each capability
- handshaking initiated for the sub protocol
    - Using status message

Wire Protocol Messages

- Hello
- Disconnect
- Ping
- Pong

Wire Protocol

Hello

- As discussed previously
- Triggers negotiating and initiating the subprotocols.

Wire protocol

Disconnect

- Disconnect from a peer
- Includes a single byte reason code

Disconnect Reasons

```
REQUESTED((byte) 0x00),

TCP_SUBSYSTEM_ERROR((byte) 0x01),

BREACH_OF_PROTOCOL((byte) 0x02),

USELESS_PEER((byte) 0x03),

TOO_MANY_PEERS((byte) 0x04),

ALREADY_CONNECTED((byte) 0x05),

INCOMPATIBLE_P2P_PROTOCOL_VERSION
((byte) 0x06),

NULL_NODE_ID((byte) 0x07),

CLIENT_QUITTING((byte) 0x08),

UNEXPECTED_ID((byte) 0x09),

LOCAL_IDENTITY((byte) 0x0a),

TIMEOUT((byte) 0x0b),

SUBPROTOCOL_TRIGGERED((byte) 0x10);
```

Wire protocol

Ping

- Simple keep alive type message (normal ping)

Wire protocol

Pong

- Response to a ping

Sub-protocols

- eth
- Light Ethereum Subprotocol(les)
- whisper(shh)
- swarm(bzz)
- …

eth

Versions

- 62
- 63
    - added 4 new messages to V62 to support Fast Sync

Eth flow

- Both nodes send status
- Protocol continues after status

Eth messages

V62

- Status
- NewBlockHashes
- Transactions
- GetBlockHeaders
- BlockHeaders
- GetBlockBodies
- BlockBodies
- NewBlock

V63 (all of V62 plus)

- GetNodeData
- NodeData
- GetReceipts
- Receipts

Status

- Inform a peer of its current ethereum state.
- [+0x00: P, protocolVersion: P, networkId: P, td: P, bestHash: B_32, genesisHash: B_32]

NewBlockHashes

- Specify one or more new blocks which have appeared on the network.
- Should only announce hashes of blocks they have and they’ve validated
- Should not include blocks which the peer knows about
- [+0x01: P, hash1: B_32, hash2: B_32, …]

Transactions

- Specify transactions that the peer should make sure are included on its transaction queue.
- Nodes must not resend the same transaction to a peer in the same session.
- This packet must contain at least one new transaction.
- [+0x02: P, [nonce: P, receivingAddress: B_20, value: P, ...], ...]

GetBlockHeaders

- Require peer to return a BlockHeaders message.
- [+0x03: P, block: { P , B_32 }, maxHeaders: P, skip: P, reverse: P in { 0 , 1 }]
- Block — starting block number to get headers from
- maxHeaders — maximum number of headers to return
- Skip — number of blocks apart from each other
- Reverse — 1 means falling from block; 0, increasing

BlockHeaders

- Reply to GetBlockHeaders
- Response is a list of block headers as per the yellow paper.

GetBlockBodies

- Require peer to return a BlockBodies message. Specify the set of blocks that we're interested in with the hashes.
- [+0x05, hash_0: B_32, hash_1: B_32, ...]

BlockBodies

- Reply to GetBlockBodies.
- The items in the list (following the message ID) are blocks, minus the header, in the format described in the main Ethereum specification. Blocks match the hashes provided in the GetBlockBodies request.
- [+0x06, [transactions_0, uncles_0] , ...]

NewBlock

- Specify a single block that the peer should know about.
- [+0x07, [blockHeader, transactionList, uncleList], totalDifficulty]
- (typically used by miners to push out newly mined blocks)

GetNodeData (new in v63)

- Require peer to return a NodeData message.
- Pass array of hashes of accounts/nodes/states to get state information for
- [+0x0d, hash_0: B_32, hash_1: B_32, ...]

NodeData

- Provide a set of values for the node data hashes from GetNodeData
- Only provide a best effort of values
- [+0x0e, value_0: B, value_1: B, …]

GetReceipts

- Require peer to return a Receipts message.
- Provide list of receipt hashes for which to retrieve the receipts
- [+0x0f, hash_0: B_32, hash_1: B_32, …]

Receipts

- Return list of TX receipts for the provided hashes
- [+0x10, [receipt_0, receipt_1], …]

Light Ethereum Subprotocol

les

- Name
  - les
- Version
  - 1

Whisper

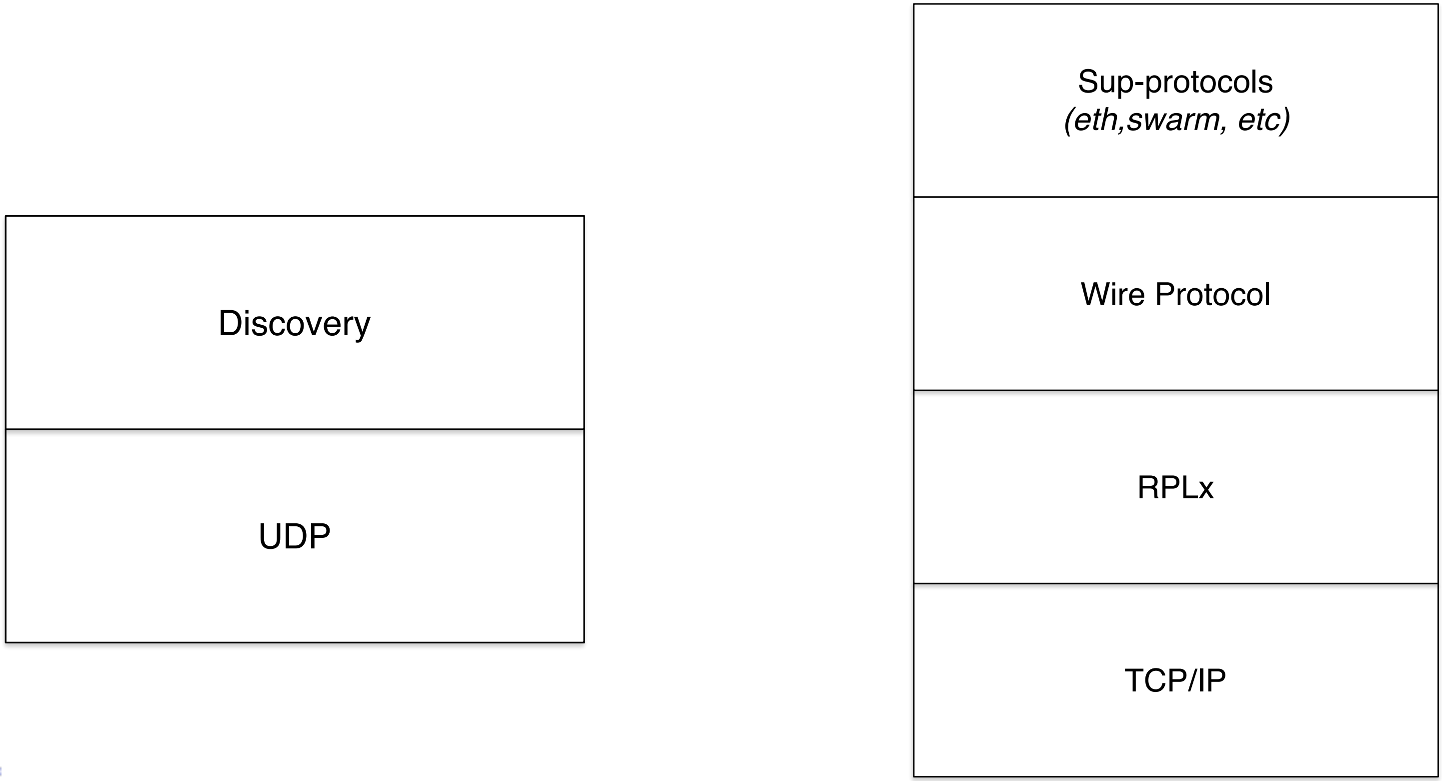
- Name
  - shh
- Versions
  - 2
  - 3
  - 5

Swarm

- Name
  - bzz
- Version
  - 0

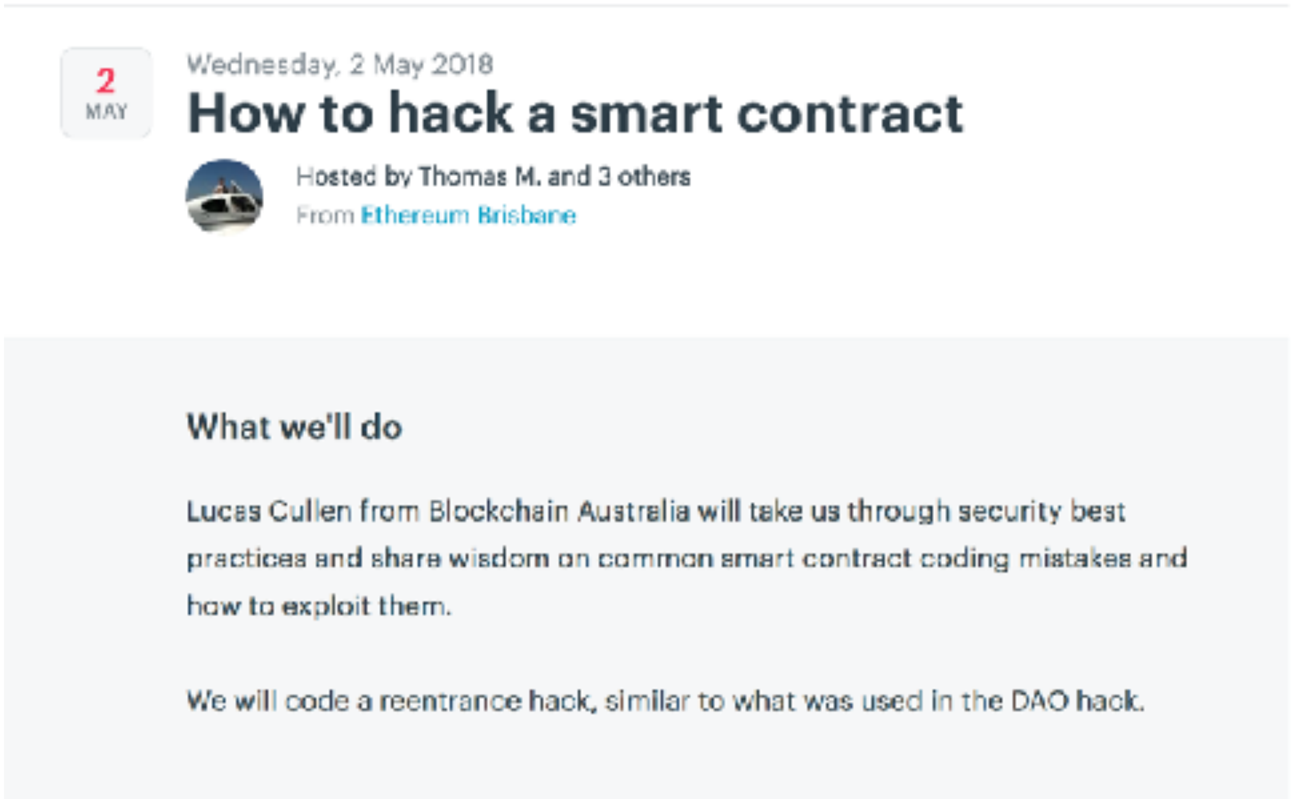
Overview

Two Protocols



Tonight

https://www.meetup.com/en-AU/Ethereum-Brisbane/events/248238988/



2 MAY

Wednesday, 2 May 2018

How to hack a smart contract

Hosted by Thomas M. and 3 others

From Ethereum Brisbane

What we'll do

Lucas Cullen from Blockchain Australia will take us through security best practices and share wisdom on common smart contract coding mistakes and how to exploit them.

We will code a reentrance hack, similar to what was used in the DAO hack.

Future

May 16, 2018	Current research from UQ, QUT, GU
May 30, 2018	Solidity Programming Language
Jun 13, 2018	Security & Cryptography Part 2
Jun 27, 2018	Consensus: PoW, PoS, PoA

Acknowledgements

- Raúl Kripalani
- Peter Robinson
- Team Anzac

Questions