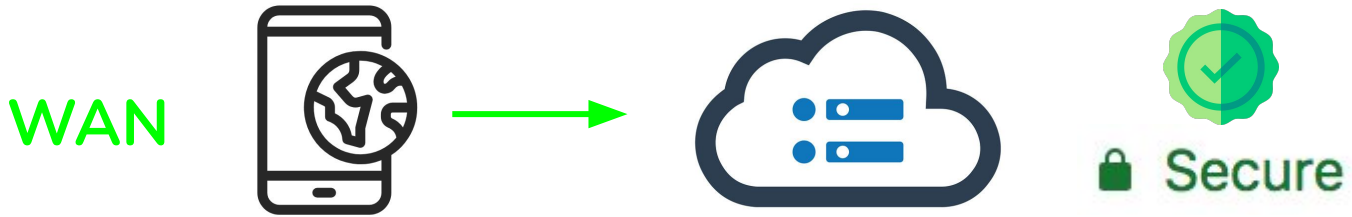
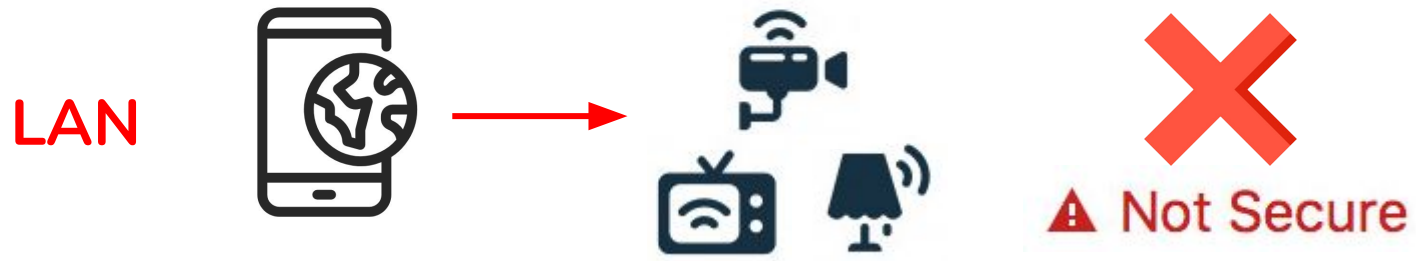
Local Devices

Building Blocks for the Local Web

What's wrong?

- The LAN is not a first-class citizen of the web/browser world





Example use-cases

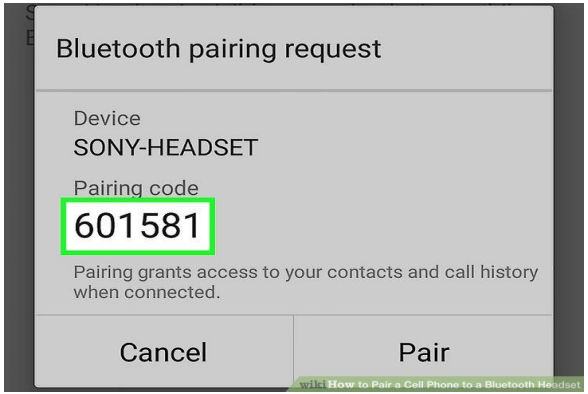
- Offline-first & Resiliency
  - Reduce dependency on the cloud
  - Should work on air-gapped networks
- Security connect to your NAS
  - Without installing self-signed certs or silly DNS tricks
- IOT in the browser
  - Space no longer reserved to apps
- LAN WebRTC signaling
  - Security camera, doorbell, ...
- High bandwidth applications
- Connect 2 browsers on LAN
  - LAN games

Many have failed

- Similar attempts
  - Network Service Discovery API
  - FlyWeb
  - Raw-sockets
  - TCP and UDP sockets
  - ...
- Cool projects, but struggles with:
  - Large scope
  - Raw/low level access
    - Breaking browser security architecture (CORS, ...)
    - Infinite list of security concerns

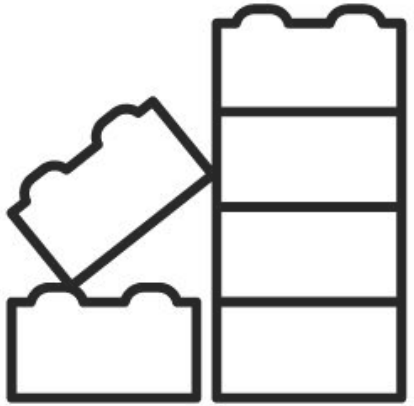
Concept

- Establish thrust by pairing
  - Network not trusted by default
  - Exchange (self-signed) TLS certificates
  - Avoid MITM attacks
- Just like casting or Bluetooth pairing!



Design goals

- LAN only
  - No NAT traversal
- Offline first
  - No cloud, no certificate authority
- Security first
  - Over freedoms such as raw wire access
- User friendly
  - Available to anyone
- Low Level Building blocks
  - Over all-inclusive higher level APIs
  - Encourage innovation
- Maximal re-use
  - Minimal new protocols and API surface



Protocol

- Same basis as OSP
  - Different protocol identifiers
- mDNS discovery
- Transport and metadata discovery with QUIC
  - Stripped down
- Authentication protocol to ‘pair’ devices
  - A paired device = Trusted Device
- Minimal, opaque Messaging Protocol
  - Send/receive binary blobs
  - Ensures at least 1 means of exchange
  - Message content: application layer (building block!)

Trusted Devices

- Easy re-connect
  - Skip pairing step
- TLS Certificates available to other protocols
  - Considered part of certificate store
  - HTTPS, WS, QuicTransport, ...
- Management
  - Globally by user Agent GUI
  - Access granted per domain

JavaScript API - Establish Connection

- Inspired by MediaDevices API

```
// Connects to a device by name. Prompts user for consent.
// If the device is not yet trusted,
// the user must first complete the authentication flow.
const device = LocalDevices.getLocalDevice({ displayName: “my_nas_server” });
```

```
// Prompts user consent to list local devices.
const devices = LocalDevices.enumerateDevices();
```

JavaScript API - Messaging

- Leveraging the WebTransport API

```
const localDevice: LocalDevice;
const webTransport = new LocalDeviceTransport( localDevice );
```

Or

```
const localDevice: LocalDevice;
const webTransport = new WebTransport( localDevice.address );
```

JavaScript API - Virtual Local Device

- Browser as virtual device
  - Enable browser-to-browser, E.g.: for LAN games

```
// Creates a virtual Local Device
// Prompts the user for permission to expose a service on the LAN.
const device = new LocalDevice({ displayName: 'my-virtual-device' });
```

Security - Fingerprinting

- Identifying user/browsers
  - For advertising or other tracking purposes

- Remedies
  - User consent required before listing devices.
    - Avoids unsolicited ‘background’ fingerprinting
  - Randomized MDNS addresses
    - Avoid leaking IP information.
  - Remove the listing API
    - If deemed necessary

Next steps

- Scrutinize design & security concerns
- Find community support base

Questions?!

References

- WICG discussion
- Draft proposal doc

Michiel De Backker

@backkem
mail@backkem.me

- Co-creator pion/webrtc
  - Pure Go WebRTC stack & API
- CTO @ twintag.com
  - Product-Led Communication PaaS

PION

TWINTAG

Take me with you!