Control of local IOT devices via secure Cloud apps.

THE NEED

IOT devices are ever more common but present several issues:

1. They are difficult to secure,
2. They can be a source of concern regarding what private information they are sending to the cloud which might be intercepted or misused.
3. They will stop working if the provider of the cloud service goes out of business or simply decides to stop supporting the product. Some level of configurability in the device to change the host address can help with that, but even doing basic configuration can be difficult.
4. They can be quite difficult to configure

At the same time, $5 micros like the ESP-8266 or RP-2040 can host web servers and provide complex user interfaces via web pages. On a local network, security is less of a concern (especially if you use CAT5 instead of WiFi) and direct access via web browser on your PC is an easy solution. Using a browser means not having to install anything. The downside is:

1. Updating the UI page on the device, but that's not so horrible, a simple file upload UI is possible.
2. If the device doesn't have a certificate (and it can't have a domain certificate) then the UI page can't access devices on the PC such as the camera, mic, location services, etc... So you could imagine a simple doorbell app that only works locally (no need for a cloud server) and allows you to talk to the person at the door, but you can't use the mic in the PC through the browser because it isn't secure.

A better solution is a web app hosted in the cloud, which can even be appcashed locally for use without internet access, which loads the UI, and then communicates with devices locally.

So I have a local (192.168, etc...) device with a web server in it. And I have a publically hosted web server with a cert hosting https web pages. I get a web page from that server, and in that web page I have javascript that tries to connect to the local device via http, https, ws, or wss as those are the only protocols supported by the browser.

THE GOAL

The Goal is a web hosted app (IDE, control panel, configuration manager, etc…), served via a certificate (https) from a public domain, that talks to local (e.g. 192.168.x.x) devices via http(s), ws(s), etc… requests from the javascript in the browser.

THE CHALLENGE

Actually making this happen is harder than it looks.

Things that don't work:

- If the local device is serving an http page or ws, that fails, 'cause browsers won't allow unsecure content on secure pages.
- Can't get a cert with an IP address as the CN.
- Can't get a domain tied cert because the local device has no domain, right?
- Can't get a real cert for a public domain (e.g. device1.mydomain.com), put it on the device, and then use the hosts file on the PC to redirect that domain to the local IP because:
  1. the browser will freak out about the domain on the cert being resolved to a local IP (probably?)
  2. modifying the hosts file is sure to trigger Anti-Virus software and be difficult for users.
- Go buy a cert. In the CN, put my real domain name. In the SAN, put e.g. 192.168.1.141, 192.168.1.142, 192.168.1.143, etc... (in case I have multiple local devices). Then put that cert on my real domain and my device at 192.168.1.141, and go hit the https page at my real domain. My public web server is gonna download javascript, and that js is gonna do an Ajax to https://192.168.1.141/, (or a wss to that address) and the browser gonna get that cert from the device, see that the SAN matches the URL, and it was issued by trusted chain, and it's gonna go, "all ok, happy happy!".
  Result: While it is technically possible to include a reserved IP address such as an RFC1918 private address in the SAN of an SSL cert, a commercial certificate authority is unlikely to offer this service as no one can claim ownership of the reserved IP addresses and as such the certificate authority could not attest to the users ownership of said IP address. Additionally, such a certificate could be used for nefarious purposes and could open the certificate authority up to unnecessary scrutiny and liability. Many thanks to Alan Copeland for testing this.
- https://stackoverflow.com/questions/67765238/mixed-content-the-page-at-was-loaded-over-https-but-requested-an-insecure-resour says that the following meta tag can be added to the HTML for the page and it will avoid the issue:
  <meta http-equiv="Content-Security-Policy" content="upgrade-insecure-requests">
  Problems: Does not appear to work in Chrome

Things we've tried that sort of work but have issues:

Current Best Solution:

- Tell Chrome to treat the device IP as safe via this flag:
  chrome://flags/#unsafely-treat-insecure-origin-as-secure
  You can enter multiple protocol and IP addresses in a comma delimited list. E.g.

http://192.168.1.142, ws://192.168.1.142
Problems: 1. Requires trust or knowledge on part of the user (browser starts with a warning message about degraded functionality), 2. Chrome specific. 3. Slightly reduces security.

Other Possible Solutions:

- Self signed cert on local device. This works, but:
  Problems: It only works after the user first directly connects, click "Advanced" / connect anyway, and you have to re-do that every so often.

- On your (Linux, Mac or WSL) PC (maybe using Powershell on Windows?) you can use SSH to re-direct the devices port to the localhost port and then set a flag in chrome to treat localhost as secure:
  `ssh -N root@192.168.1.142 -L 443:192.168.1.142:443`
  chrome://flags/#allow-insecure-localhost
  and then you can talk to ONE robot. Or you can redirect to multiple local ports and each robot is one port.
  Problems: 1. Not fully tested yet, 2. Requires extensive trust or knowledge on the part of the user, 3. Really reduces security. 4. Limited to certain OSs?

- Rhett Garber says: "I've had luck using "ngrok" (https://ngrok.com) to temporarily make an internal ip / web app public at an anonymous address. Only downside is that you're relying on the free tier of a commercial company."

- Open an iFrame with the link to the http site and then use:
  https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage
  to communicate between the main page and the iframe page.

Ideas:

- Someday browsers will support raw sockets? Chrome does, but only in a few select cases.
  https://wicg.github.io/direct-sockets/#sotd
  https://chrome.google.com/webstore/detail/secure-shell/iodihamcpbpeioajjeobimgagajmlibd

- Install a small proxy server on your local computer that translates requests from the device into local host http or ws traffic and do the allow-insecure-localhost thing above.
  Problems: Same as the above, but would allow direct access to raw sockets on the device.

- Just an interesting note:
  https://stackoverflow.com/questions/60448948/mixed-content-request-from-https-page-to-http-non-https-localhost-address-not
  If you get one page via https, then you can get pages via http from the same IP.

THE QUESTION

Do you have ideas on how better to do this?