ALGORAM

An Operating-System-Independent Web Front Panel for Radio Transceivers

- Bruce Perens K6BP
- bruce@perens.com
- +1 510-4PERENS (+1 510-473-7367)

Demo

- Connect to WiFi or use your cellular connection and run Chrome, Firefox, or Opera.
- server1.perens.com runs audio echo server in my home in Berkeley, on Comcast Business cable Internet service.
- server1.perens.com/canvas.html runs the spectrum display.
- Blog.algoram.com has the authentication demo.
- iOS won't do the audio echo.

What Is It?

- Algoram has produced a web front panel that will run on all popular operating systems except Apple's iOS.
- This is not ported from one system to another. The exact same software runs on Windows, MacOS, Android, Linux, BSD, Kindle Fire, and other systems.
- It uses two-way audio through the web browser to provide a microphone and speaker for the radio.
- It runs well on smartphones and tablets, as well as laptops and desktops.
- No installation is necessary.

The Server

- The server to enable this is written in C++, and runs on small processors without MMUs, as long as the they can support TCP and a network interface.
- On Algoram Katena, there is a WiFi interface on the radio which you can use with a smartphone.
- There's also an Ethernet connector.
- The same front panel works locally, or globally on the Internet.

Authentication

- There is authentication software that can authenticate a Stranger on the Internet as a licensed radio amateur, solving the “don't touch my knobs” problem for the Internet.
- Authentication through an approved user list could be implemented.

Dexterity and Handicaps

- The Smartphone interface is usable by fumble-thumbed people with relatively low dexterity.
- Keyboard inputs are recognized.
- Visually impaired users can use the keyboard for input exclusively rather than the touchscreen.
- Motor-impaired users can use any device that emulates USB key presses.
- We could implement voice output for blind users.

What are the Breakthroughs?

- The key to this development has been emerging HTML5 APIs:
- We can use the microphone using the getUserMedia() API.
- We can do sophisticated audio processing using the Web Audio API.
- We can do essentially any graphics, 2D or 3D using the HTML5 Canvas.
- We communicate with Websocket.
- Web browsers support real-time compilation of Javascript, so it's fast.

Platform

- The current platform is the Algoram *Katena* radio. The program is working as a WiFi host which communicates with smartphones, etc.
- The software would work on other platforms that combine a radio, a server computer, and a net connection, such as Northwest Digital or FlexRadio.

Revenge of the Clones

- Wouldn't it be nice if we all ran the same operating systems, instead of Windows, Mac, Android, iOS, Linux, BSD?
- Well, lets look at other situations where everything's the same.
- This is Jango Fett, the clone trooper. Would a clone army be a good idea from a strategic perspective?

Clones Today

Commercial Bananas are Clones

- Commercial Bananas are bred to be seedless and are propagated by cuttings. The are true clones.
- The *Gros Michael* cultivar of my parents time was sweeter than the Cavendish banana we eat today.
- But Gros Michael fell victim to Panama Disease, a fungal infection which spread worldwide.
- The “Tropical Race 4” version of Panama Disease is almost certain to eliminate the Cavendish banana eventually.
- Clones share the same disease immunity and all of them can fall victim to the same disease.

Tasmanian Devils are Genetically Identical

Tasmanian Devil Disease

- Tasmanian Devils (Tassies) are inbred and went through a genetic bottleneck, so that they are all genetically identical and share the same immune complex.
- They have contagious cancer which is driving them to extinction.
- Humans don't have contagious cancer because they are not genetically identical.
- If you have any cancerous twins, don't bite them! Tassies do that and spread the cancer between themselves.

Computers

- 15 years a go, a big international express shipping company had standardized on Microsoft Windows across its entire operation.
- Then, they got the Red Flag virus.
- They were reduced to phones and fax machines across their entire operation for a day or more, until they would bring all of the Windows systems down and rewrite their software.

So

- Clone armies are a really stupid idea. When Jango and Boba Fett catch a flu, the whole army catches the flu. Sorry George Lucas.
- Similarly, if we all use the same operating system, all of our computers can catch the same virus at the same time.
- Ham Radio is the public service network of last resort. It would be bad for us all to run the same software. It's bad for the world to do that, too.

Hetrogenous Networks

- Hetrogenous Networks have different software running on their nodes, instead of a monoculture.
- This is important for security.
- But there is a cost.
- They don't run the same applications.
- We have to “port” applications to each platform.

Porting is Expensive

- There is a combinatorial problem in supporting all of these different operating systems.
- Small companies like Algoram can't afford the programmers to support all of those different operating systems. Right now we're just two guys who work part time.
- Programmers have tried different strategies to reduce the cost of porting.

Porting Strategies

- We can use the same programming language on different platforms. For example, C and C++ run on almost everything.
- But even then, platforms have different graphical user interfaces, etc., and thus applications for each operating system have to be programmed differently.
- Portability Layers like wxWidgets and the Java API can be used to hide the differences between platforms.

Software as a Service

- Run everything on a web browser, with someone else taking over the effort of maintaining the software (for a fee, an advertising opportunity, or the opportunity to know your data).
- This tends to fail in a disaster, and spread the impact of the disaster beyond its boundaries. For example Hurricane Sandy.

The Web Browser as GUI

- Modern web browsers, meaning Google Chrome, Firefox, and Opera, provide all of the APIs necessary to be radio front panel.
- We have three different browsers that run the necessary APIs. They are different internally, and so they probably won't all catch the same virus at the same time.
- Safari and Internet Explorer will eventually catch up, giving us five browsers. Maybe others eventually.

Suported Platforms

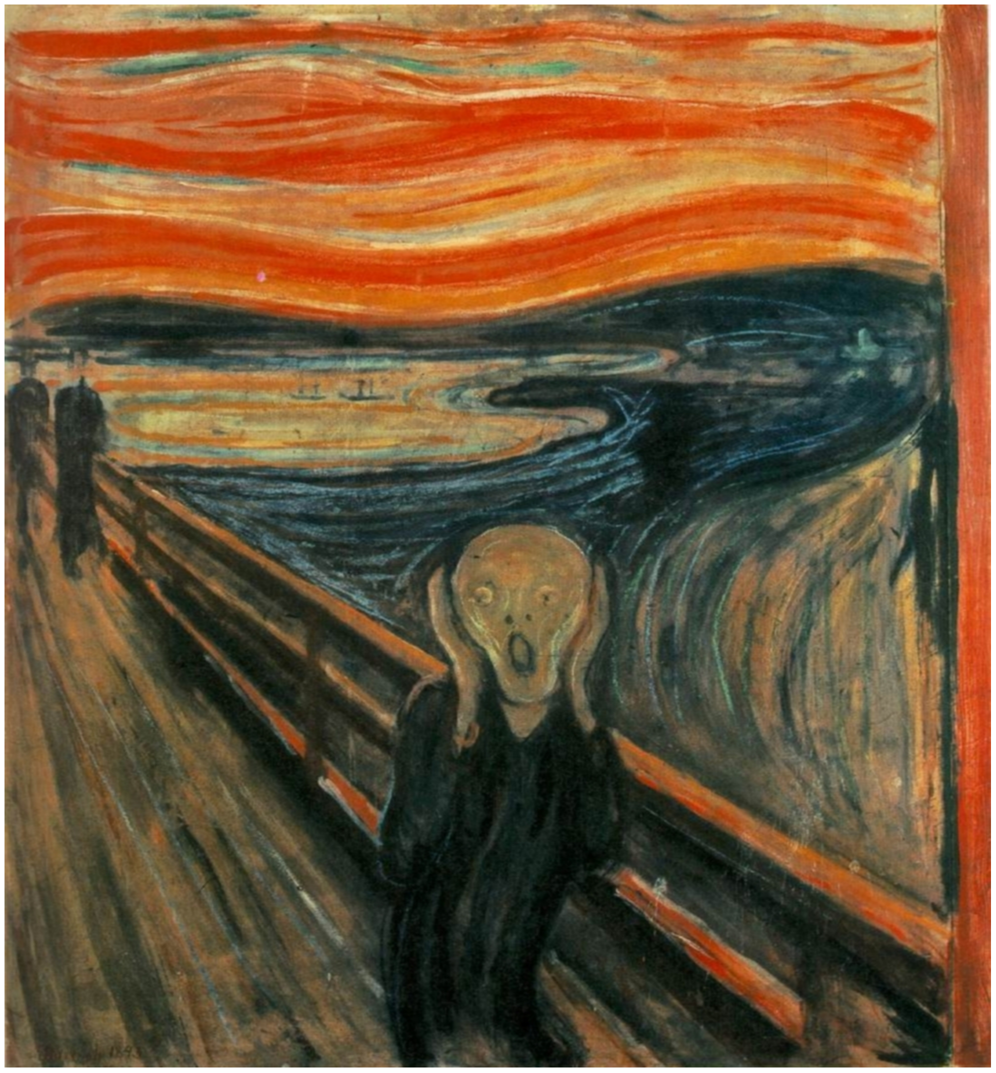
- At present, we can run the exact same software on one of those web browsers on Windows, MacOS, Android, Chromebooks and ChromeOS, Linux, BSD, Kindle Fire, and any other platform that runs those browsers in their full form.
- That's not iOS. Not even iOS 9.

Why not iOS?

- Apple requires that all browsers on iOS run Apple's version of the Webkit rendering engine, instead of their own.
- Apple has not kept up with emerging web APIs. No complete implementation of getUserMedia() or the Web Audio API.
- Because their web store is so locked down, you can't fix the browsers. Your only choice is to write a specialized app for their platform.
- That's why you pay more for Apple hardware.
- But there is one advantage to Apple hardware:

Adantage of Apple Hardware

- The iPhone and iPad charging cable doesn't fall out.
- This justifies the up to 3X price increase over similar Android hardware, the lock-down, and the obsolete software.
- However, Macbook has already switched to USB C...

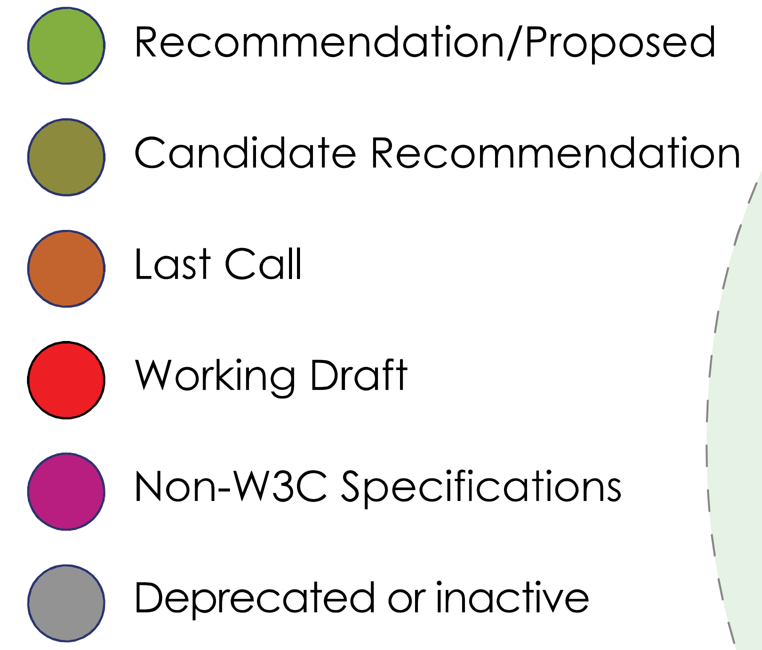


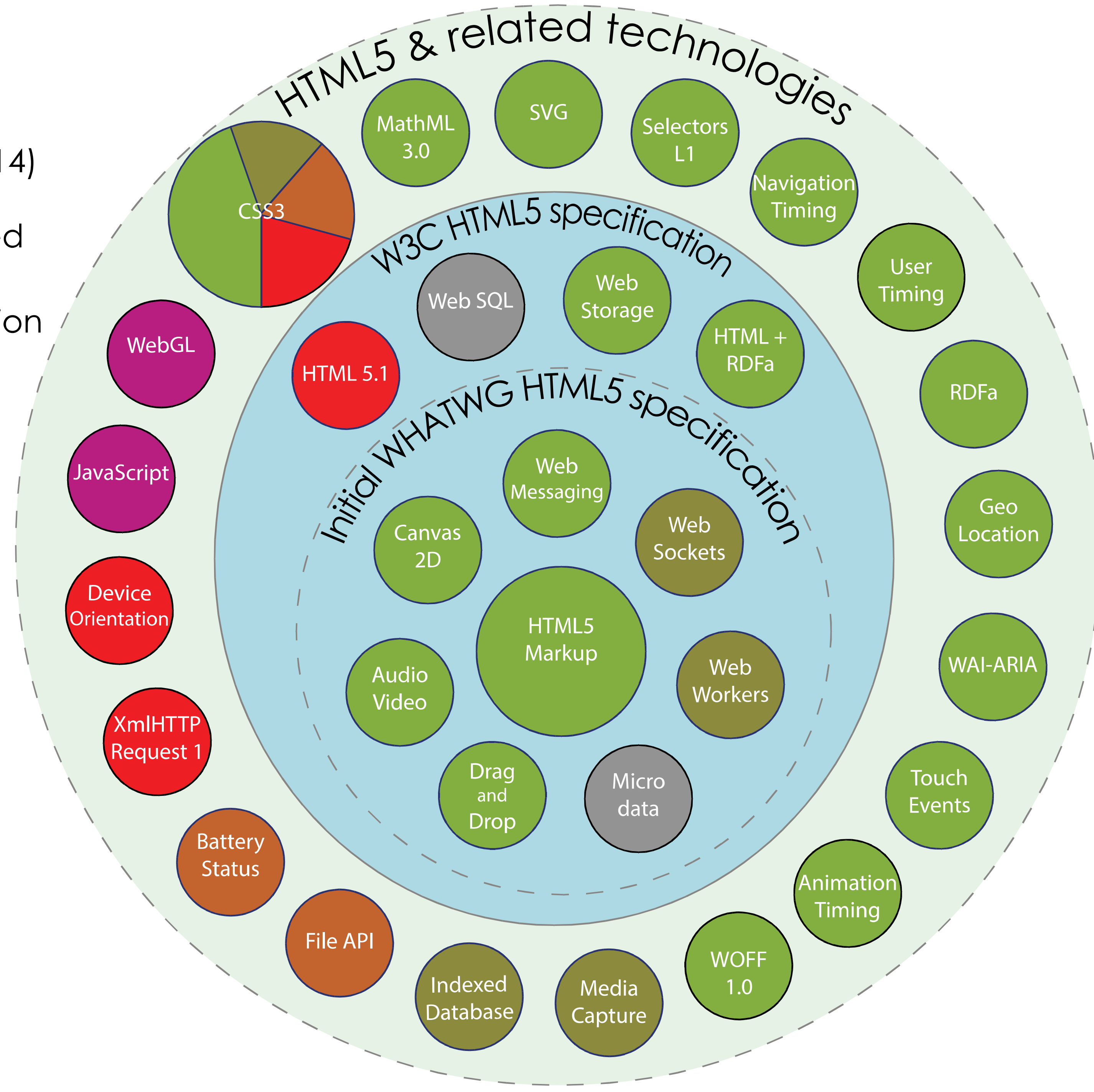
Original Web Browser Functionality

```
<head>
  <title "Hello World!">
</head>
<body>
  <p>
    Hello, World!
  </p>
</body>
```

HTML5

Taxonomy & Status (October 2014)





Assumptions

- Amateurs always plan for their systems to survive the apocalypse.
- Thus, no software-as-a-service in the conventional model.
- In a disaster, it's the services that fail.

Local Server

- The server runs in your radio, and stays up during a disaster.
- You aren't dependent on anything outside of the radio to run your front panel. No network necessary.
- Although you may not have access to the Internet, you probably do have access to a radio network, even if it's an ad-hoc one.

Web Apps

- A new API, supported on Android and iOS, allows web pages to become full-screen apps.
- You can install them as apps, or just save them from the browser. But they are really just web pages.
- They get rid of the URL bar and all of the other web server interface elements.
- They look like any other app.

Start-Up (1)

- Connect to radio via WiFi or Bluetooth.
- Radio runs DHCP server.
- DHCP sends route to its local network and possibly Hamnet, *without* a default route for the entire Internet. Phone keeps its old Internet route working.
- Open Radio URL in browser (or just push app button to do this).
- Radio runs DNS to send you its local network address.
- Connect to radio web server.

Start Up (2)

- Radio Web Server sends HTML file.
- HTML file causes browser to request Javascript and CSS file.
- Radio sends those.
- Javascript runs.
- A two-way, stream-oriented connection to the radio is made from Javascript using Websockets.

Start-Up (3)

- Javascript gets Microphone (and possbly camera) via getUserMedia()).
- Browser asks permission to use microphone, if that isn't saved.
- Application starts.
- All data communicated via one Websockets channel.

Local or Remote

- The same protocol works on a local network or remotely over the Internet.
- We solve the “Don't Touch My Buttons” problem:
- For local clients, the WiFi network (which is Part 15, not Amateur Radio) is password-protected and encrypted (very easy for user).
- There is authentication for remote Internet clients via HTTPS and x.509 Cryptographic Certificates (this is not going over Amateur Radio). Certificate management adds steps the user has to go through.

Communications Protocol

- My original assumption was that we'd use WebRTC.
- However, there is no *small* embedded library that supports WebRTC on the server. The smallest implementations are based on the Chrome browser. For example Node.js, a system for running Javascript on servers, is too big for our hardware.

Websockets

- So Websockets was chosen as the communications protocol.
- Websockets is a stream-oriented protocol on the HTTP or HTTPS socket. It gives you the session-oriented operation that we have previously had to simulate on the web using cookies. It knows how to serialize and de-serialize Javascript strings and arrays.
- It traverses firewalls the way any web browser connection does. If web connections get through, this will too.

JSON

- JSON is JavaScript Object Notation.
- It is a way of putting structured data into a string format (serialization) that is easy to parse by computers and human readable.
- It handles strings, numerics, booleans, data structures and arrays of those things, including hierarchical ones.
- Javascript in the browser has native functions to serialize and parse JSON.
- Lots of software, in many languages, is available to parse and produce JSON.
- It works well for both intercommunication and a file format.

Websockets Data

- Websockets knows how to send and receive strings, arrays, and “blobs”. Strings and arrays are easy to handle in Javascript, blobs are meant for raw data not in a Javascript format, so they need extra effort.
- JSON data is converted to and from strings for transmission. All sent and received strings are encoded JSON.
- Audio and panadapter spectrum display data are arrays.

JSON Use

- All small data is communicated using JSON, this is mainly commands to the radio and status back.

We send: { “command”: “transmit”; };

We receive: { “command”: “transmit”, “frequency”: 144.52, “mode”: “FM”, “deviation”: 15000, “PL”: [100, null] }

- Commands and status look the same.
- This is really easy to write in Javascript code to run on the browser, and not much worse to write in C++ on the server.

Bulk Data

- Audio and the “panadapter” spectrum display are bulk data, are sent as Javascript arrays.
- The first number in the array says what the type of the data is.
- Currently that is only audio and spectrum data.
- Everything else is sent separately as JSON.

Audio

- The current audio implementation just sends 16-bit samples at 8192 samples per second, without compression.
- This works perfectly on the local network, no compression necessary.
- It's sub-optimal on the Internet, but mostly works fine.

Codec2 In The Browser

- A program called *emscripten* compiles C to Javascript.
- It's a Javascript back-end to the LLVM compiler.
- The just-in-time compiler for Javascript in your browser has enough performance to run C code acceptably.
- So, it's probably possible to run Codec2 in the browser, portably.
- Long-term, this is what WebRTC is meant to do, and we will switch to it when we have a good embedded library for it.

Web Audio API

- The web audio API provides a graph-oriented programming interface for audio processing with many interesting processing nodes.
- There is a compressor/limiter node.
- There is an FFT node! Yes, FFT is an emerging standard API implemented in browsers.
- Coupling Javascript to web audio processing is awkward. Stream to array and back at inputs and outputs, separate processing stream each time you go back and forth to Javascript.
- Ultimately we're going to WebRTC, which will run its own codec for us.

Sideloading

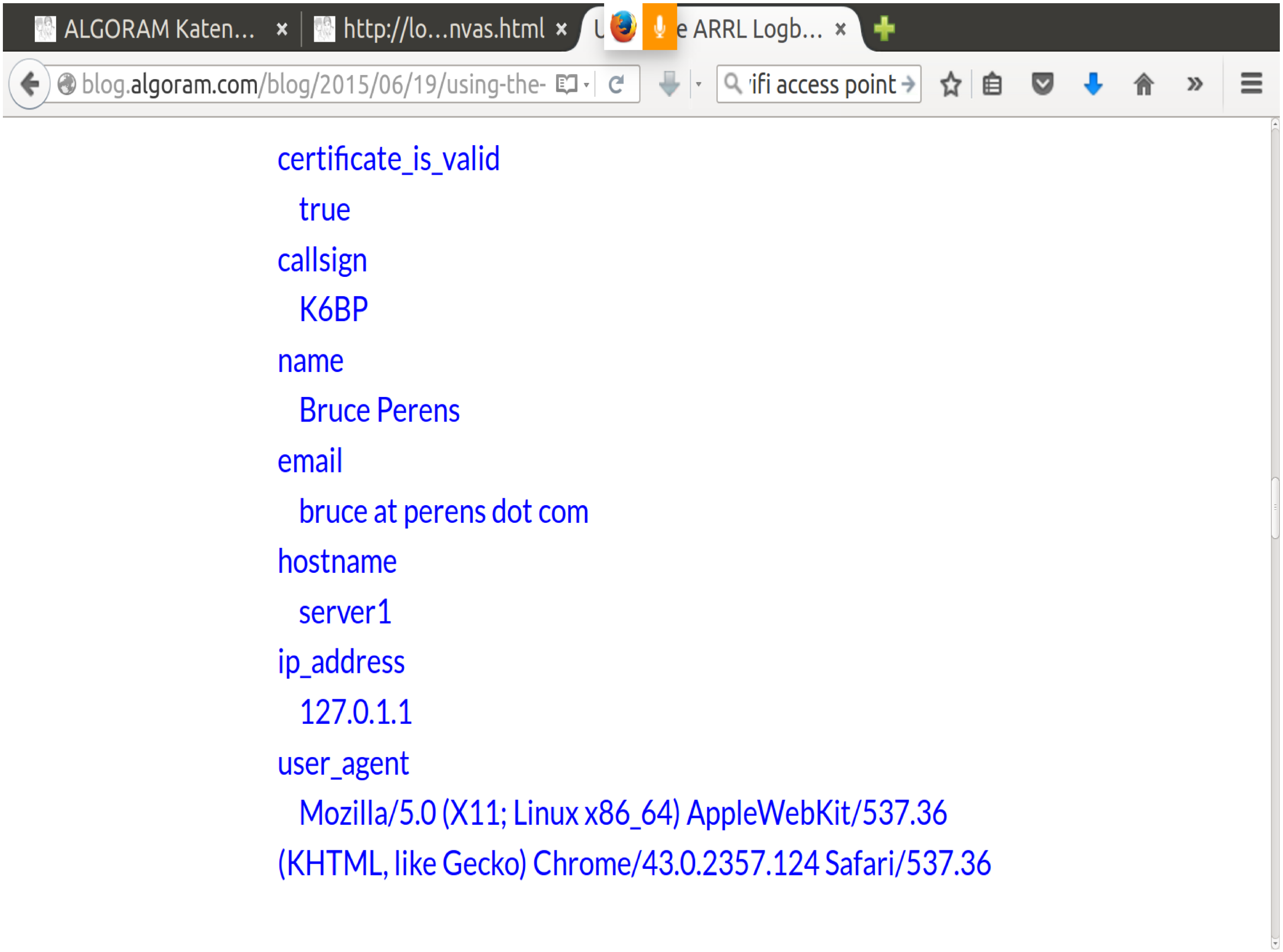
- For Kindle Fire, it was necessary to use *sideloading* to install Chrome instead of using the Silk browser. It was not, however, necessary to root the device.
- Rooting means defeating the security of the device.

Authentication

- Two years ago, Heikii Halikaanen presented how to use the Logbook of the World cryptographic certificate to validate Strangers on the Internet as licensed Radio Amateurs.
- The Trusted QSL program can be used to extract an x.509 cryptographic certificate that your browser can use for a secure, authenticated connection.
- The certificate includes your callsign.
- Instructions on blog.algoram.com .

Digital Certificate

- ARRL has already authenticated many thousands of Radio Amateurs. Anyone on LotW can use this.
- It's really cool that they do the work for us.
- We would need a table to correlate callsigns to nations and license classes, to tell what people really are licensed to operate.
- You can also password authenticate individuals.



certificate_is_valid
  true
callsign
  K6BP
name
  Bruce Perens
email
  bruce at perens dot com
hostname
  server1
ip_address
  127.0.1.1
user_agent
  Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.124 Safari/537.36

ALGORAM