EAP Onboarding for WiFi devices

Michael C. Richardson
Sandelman Software Works Inc.
mcr@sandelman.ca

January 20, 2023

1 Executive Summary

WiFi IoT devices and laptops currently are difficult to onboard to WiFi networks without a human to push buttons and do configuration. The use of insecure networks is often required.

This document is about running code that will enable IoT devices to onboard easily, and also allow laptops to transparently join encrypted networks.

This work could also easily allow many unencrypted guest networks that employ captive portals to be encrypted in an friction-free way.

2 Problem to be solved

In many environments an unencrypted wifi must be deployed in order to allow guest access. The guest machines will face a captive portal, and the better ones will use the IETF CAPPORT mechanisms to signal the portal's existence. But, the underlying wifi remains unencrypted, offering all sorts attacks involving spoofing ARP/ND entries for the first hop routing and then hijacking traffic.

While some guest network access is provided via the well-known WPA passphrase posted on a blackboard along with the days' specials, until WBA3 this was also subject to attack, since the local "secret" is available to every attacker.

The use of EAP-TLS, also known as WPA-Enterprise, alleviated all these problems, but in general, it requires some kind of unique credential to be deployed to all devices. This is in stark contrast to the desired guest access that many locations would like to support. Some events, such the IETF, deploy EAP-TLS with a well-known username/password of "ietf"/"ietf", but even in those situations well educated experts often find it challenging and confusing to pick the right parameters.

In the IoT space there are multiple ways to do onboarding. The original DPP proposal suffers from requiring unique access to public 802.11 frames that is often not provided by existing device firmware. EAP mechanisms are often described as brittle, slow (one outstanding packet), and are limited to 64K of data transfered. Running DPP, or BRSKI (RFC8995), or another protocol over ethernet would be significantly easier.

BRSKI can be used to onboard new devices with the right credentials for then doing EAP-TLS, as can future versions of DPP, and even MATTER's onboarding protocol results in credentials that could be used for EAP-TLS.

For IoT devices that lack user interfaces an automated protocol is essential, but even smartphones and laptops benefit from moving to EAP-TLS. In particular, the WPA-PSK is doomed, and needs to be replaced, but replace it requires automation. The problem with WPA-PSK is that it's not device specific, and while one can make device-specific PSKs, they depend upon having deterministic and constant MAC addresss, and the Randomized and Changing MACaddress (RCM) systems break this. This is the topic of the IETF MADINAS WG.

The document draft-richardson-emu-eap-onboarding proposes a simple mechanism to enable BRSKI (and other protocols) to operate easily with current WiFi network architectures. Not only do they turn WIFI into a useable onboarding LAN, they also support smartphone and laptop uses where a captive portal will be providing final authorization.

A criticism of the dedicated WiFi SSID for onboarding is that it will be hard to deploy for Enterprises. What? Yet Another SSID? But our access points are already falling over. The difference is that the EAP-TLS mechanism proposed here works exactly with the current quarantine and guest networks that Enterprises *already* have.

Even for typical hotel/coffee shop hot-spots, where there will be a captive portal, today's state of the art is that the SSID is unencrypted. This single new encrypted SSID will rapidly replace this unencrypted SSID, as the new SSID is

easily identified as the designated guest network in the beacons.

This proposal is about getting the running-code part of the IETF process working, so as to better inform the running consensus part.

3 Proposal

This document does not repeat the technical proposal outlined in draft-richardson-emu-eap-onboarding. It addresses only the work that must be done.

3.1 Freeradius server

The freeradius 3.2 server base code needs to be extended to support the authentication-less EAP-TLS method. This requires new code paths, plus testing against regression, plus then positive testing of the new code paths.

Freeradius is the pre-eminent open source radius server, and it used extensively within 3GPP operators, as well as eduroam.

Freeradius test infrastructure will need to be extended to support no-authentication EAP-TLS mechanism in order to validate code going forward.

3.2 wpa_suppliant

The open source wpa_supplicant is the most common supplicant in use, and it needs to be extended to support this new method.

In addition, it needs to support searching for a WiFi SSID with the required eap.arpa realm, and automatically selecting it.

The Microsoft, and Apple OSX and iOS operating systems do not use the same code base, so it will fall upon those companies to provide patches.

3.3 Android Open Source Platform (AOSP)

The Android Open Source Platform code base contains a supplicant, and a proof of concept will be made to have the device scan and recognize eap.arpa realms as guest networks, in the same way that it recognizes unencrypted wifi.

This will result in a Gerrit patch set which perhaps Google will consider merging. The code will otherwise have to be tested on a compatible AOSP phone, such as a Pixel or other unlocked phone.

3.4 hostapd

The open source hostapd tool is a very common authenticator found in openwrt, and patches may need to be made to hostapd in order to allow it to announce the new eap.arpa realm.

It is hoped that no patches will be necessary, but there is still a need to work with the upstream openwrt configuration system in order to make turning on this new SSID trivial.

3.5 FreeRTOS wpa supplicant

The FreeRTOS microcontroller operating system, which is ubiquitous in Adafruit, Amazon and other IoT ecosystems contains a supplicant, and it needs patches to do EAP-TLS without authentication.

Upstreaming of this patch is critical to getting this new mechanism deployed in the IoT community.

It also needs to be able to search automatically for the right EAP Realm to join.

3.6 Virtual Testing

To the extent possible the code will be tested using virtual networks and virtual machines. There may be additional test benches requires to make this work.

3.7 Physical Testing

As with all things involving radios, physical testing with real parts will be necessary. This will be done with Omnia Turris access points with hostapd, Freeradius servers running as virtual machines and under openwrt, and with ESP32 wifi-enabled microcontrollers running FreeRTOS and RIOT-OS.

4 Deliverables

There will be a number of upstream pull requests:

1. Freeradius server updates.
2. wpa_supplicant changes
3. hostapd changes
4. openwrt configuration changes
5. FreeRTOS changes to supplicant
6. RIOT-OS changes to supplicant
7. Gerrit pull request for AOSP

5 Schedule and timeline

It is anticipated that this work will be completed before the end of 2023.

1. setup freeradius, hostapd, FreeRTOS with EAP-TLS, but without onboarding@eap.arpa.
2. do testing client for freeradius with EAP-TLS w/o client authentication
3. augment FreeRTOS supplicant with code for no-authentication
4. teach hostapd to insert eap.arpa realm
5. augment FreeRTOS with search for appropriate realm
6. augment wpa_supplicant with no-authentication and search for appropriate realm
7. setup AOSP on physical phone, test against beacons, implement patch to support onboarding@eap.arpa
8. do end to end system level testing of all components

6 Cost

This proposal is asking for 40,000USD in support from the Comcast Innovation Fund. The money will be spent as follows:

1. 20 days effort @ 500USD/day work on Freeradius and wpa_supplicant
2. 20 days effort @ 500USD/day work on hostapd, and FreeRTOS
3. 10 days effort @ 500USD/day work on AOSP and Android testing
4. 20 days effort @ 500USD/day on integration and system level testing
5. 10 days effort @ 500USD/day dealing with reviews on upstream pull requests