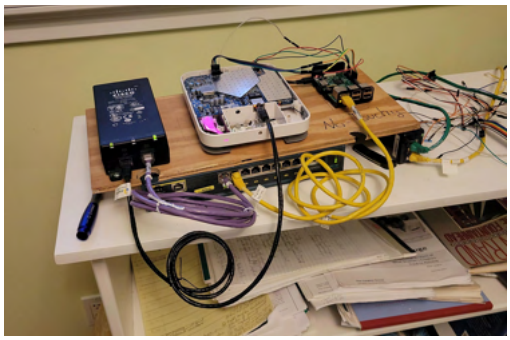
WiFi Tracking System - Technical Takeaways

Charlie Sands - January, 2022 - Northville, Michigan

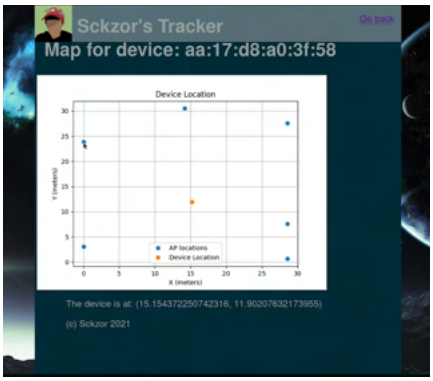
Overview



The flashing jig I built in order to jailbreak my wireless access points

For my International Baccalaureate Personal Project I wanted to track the locations of the people in my school. I reasoned that the most effective, cost friendly way to accomplish this was to analyze the relative WiFi signal strength of their mobile devices at multiple points and trilaterate the location in the room. I bought six Meraki (Yay, RoofNet!) MR18 wireless access points. These access points are cheap, simple and, due to a software vulnerability, easy to flash with custom firmware. I connected my Raspberry Pi 3 to the JTAG debug header of the device and used the Open On-Chip Debugger to interrupt the boot process and reflash the onboard memory. I installed OpenWRT, a Linux-based operating system for network appliances. I developed software that places all but one of the access point's radios into "monitor mode" and records as much network data as possible. I set the remaining wireless radio to transmit the data to a processing server. Based on the relative signal strengths from 3-6 access points the processing server determined the location of the device. The calculated device location data was sent over a cellular uplink to a web server where the results were displayed.

Rewriting the Boot Flash



A map generated by my system showing the location of a mobile device

The boot sequence and JTAG of the Meraki MR18 are not protected. After the hardware is initialized by the bootloader it is possible to stop the boot procedure of the device using a breakpoint set over JTAG. With the execution stopped, I overwrote the part of RAM that contains the factory Linux image the processor was about to load. When I resumed instruction execution the processor loaded and executed a minimal Linux image copied into RAM rather than the factory image. With the access point bootstrapped into a minimal rooted Linux environment it was possible for me to bring up the network stack and copy a full version of OpenWRT into the boot flash of the device from an HTTP server. After rebooting the access point would now be running OpenWRT instead of Meraki's custom version of Linux. Chris Blake's (riptidewave93) work was instrumental in performing the code injection. I had to modify his procedure to work with my flashing setup, however, his analysis revealed the proper place to stop code execution and the correct memory address for loading the initial and permanent Linux images. He was also responsible for compiling the initial Linux image used to bootstrap the installation.

System Network Architecture

When an access point is connected to my network, it sends self identification data to the aggregation server to inform it of its physical location. The aggregation server adds it to a directory of known receivers. The access point begins sending packet monitoring summaries, in JSON format, to a REST API running on the data aggregation server. Each discovered device and all of its signal strength information is added to a queue. A separate trilateration program sends an API request to retrieve signal strengths in the queue. Based on the directory of receivers and signal strength information an approximate location is calculated and the old data is discarded. The web frontend makes an API request for the devices and their locations, returning it to their browser.

One of my MR18s with the rear shell removed

Approximate Cost	$100
Approximate time spent	6 months
Skills developed	- Hardware Penetration Testing and Hacking - Embedded Software Development - Computer System Network Architecture - Efficient Transmission of Big Data