# WUDS: Wi-Fi User Detection System

Thursday, October 2, 2014

## Update

Monday, October 6, 2014

Several weeks ago I attended BSides Augusta. If you haven't been to the Augusta version of Security BSides, you're missing out. I've been attending BSides conferences for several years, and none have come close to the atmosphere, quality, and comfort of BSides Augusta. It's a classy deal all around in a city that supports it's InfoSec community. Well done Augusta.

So...

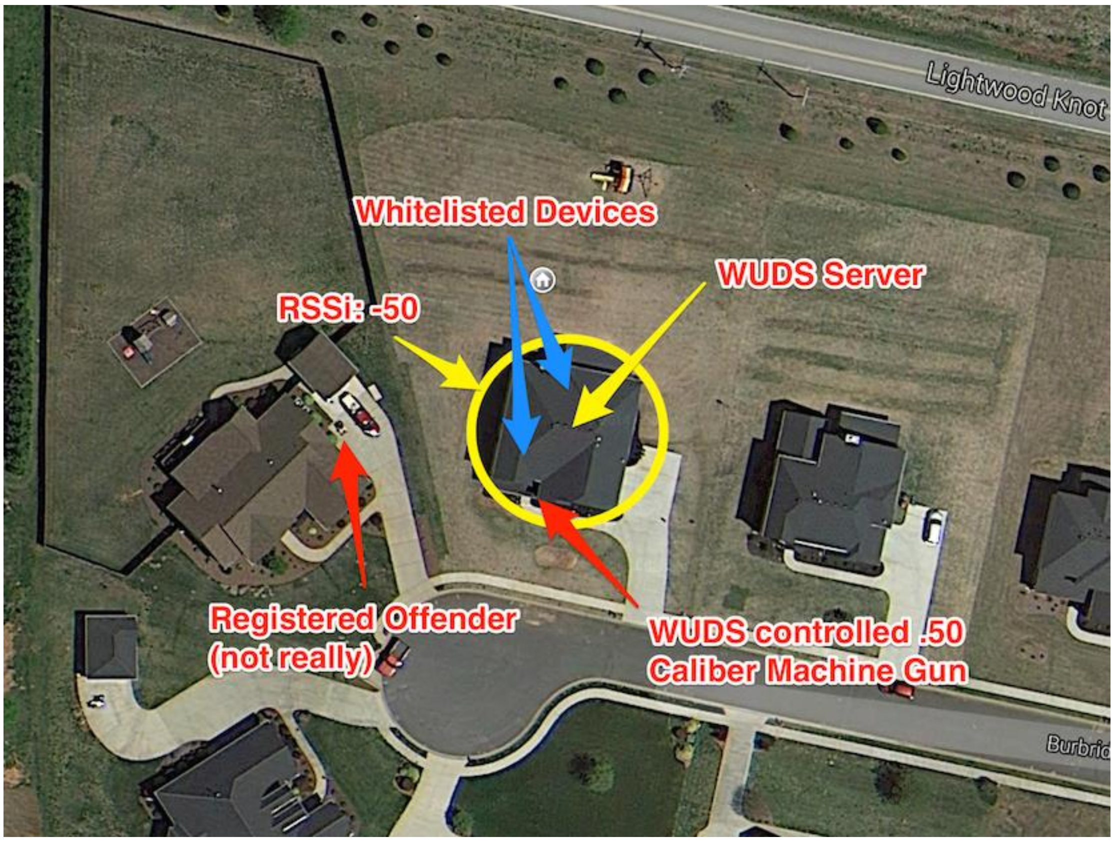
## History

I attended a BSides Augusta presentation by Tim Fowler (@roobixx) called When Zombies take to the Airwaves. During the talk, Tim discusses some really innovative ways that humanity could use Wi-Fi to track zombie population movement and communicate emergency instructions in a post apocalyptic environment. As Tim was speaking about using Wi-Fi probe requests to track infected cell phone carriers, the guy next to me, Ryan Wilson (@RyanWilson57), says something along the lines of, "You could use that same concept to alert you when zombies were approaching your house." And the idea for WUDS was born. Ryan also later named the tool. Thanks Ryan!

## Technical Approach

Wi-Fi probe requests are a great source of information from which to detect the proximity of Personal Electronic Device (PED) users. Just about everyone nowadays carries a mobile phone, which is essentially a handheld computer with multiple means of data transmission. One means of transmission, which is enabled by default and rarely ever disabled, is Wi-Fi. Most of my readers are aware of this, but many people don't know that when a Wi-Fi connected device user walks out of range of a connected Access Point (AP), the device's Wi-Fi adapter begins to make probe requests, looking for APs that have been previously connected.

The application of this technology in a physical security context is simple. The combination of a white list of unique identifiers for devices that belong in the area (MAC addresses) and signal strength (RSSI) can be used to create a protected zone. With tuning, this creates a circular detection barrier that, when crossed, can trigger any number of alert systems. WUDS includes an SMS alert module, but the sky is the limit.

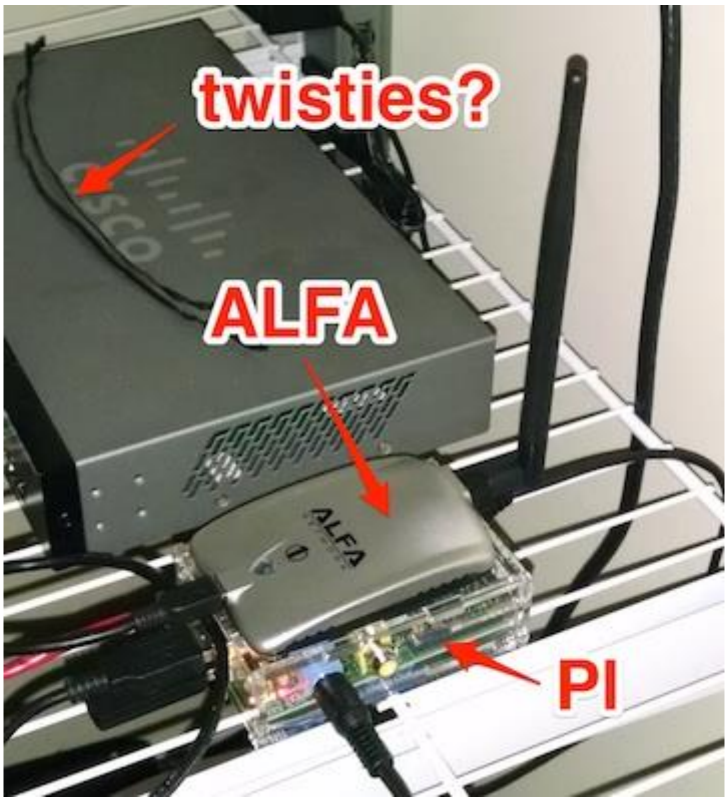

Lightwood Knot
Whitelisted Devices
WUDS Server
RSSi: -50
Registered Offender (not really)
WUDS controlled .50 Caliber Machine Gun
Burbrid

Now, when I say everyone carries a mobile phone, that includes criminals. In addition to alerting on the presence of foreign Wi-Fi devices, WUDS logs time stamps and probe request packet data. Having the time stamp and MAC address for all devices in an area during criminal activity can be quite useful. Law enforcement can take a MAC address to the cellular SP where the owner of the device can be identified. And not only can the MAC be used against a perpetrator, but probe requests also contain the ESSIDs of the networks that have been previously connected. Therefore, if a probe request is seen with the SSID of "Bag-Guy Wi-Fi", something like Wigle can be used to geolocate that SSID and provide additional attribution. Good luck avoiding arrest and prosecution when I can place your device at the scene of the crime during the time that the crime took place and prove that it was your phone through ISP records and a PushPin dropped on your house. Pwned.

Setup

Applying this concept in a densely populated area has it's challenges, as the white list and RSSI threshold must be highly tuned. In a rural area, similar to where I live, the system functions spectacularly. I accidentally left the system on the other day after I had been doing some testing the night before. In the middle of the afternoon the following day, an SMS message came through to my phone notifying me that someone had crossed my detection barrier. I was about to write it off as a false positive when the doorbell rang. It was a delivery service dropping off a package.

For my particular setup, I decided to use a Raspberry Pi as my WUDS server; the same Raspberry Pi that controls my whole home audio system. The wireless card I use is an ALFA AWUS036H connected via USB.



Guidance for installing and configuring WUDS is posted in the repository. A link to the repository can be found over on the Projects page.

Implementation Issues

For those that like looking at code, you'll notice that I am NOT using Scapy to sniff for Wi-Fi probes. Originally, I had been using Scapy, but was experiencing 100% CPU utilization while the system was running. I spent quite a bit of time trying to resolve the issue when I determined that using a raw socket and parsing the Wi-Fi packets myself would probably be a better approach. As it turns out, I was able to write my own parser using primarily builtin Python modules. For those that know me, you know that I really dislike relying on 3rd party dependencies, so this pleased me. Plus, it allowed me to avoid installing Scapy and all of it's dependences. The custom parser uses less than 1% CPU utilization on average. That is quite an improvement over Scapy.

I owe Joff Thyer (@joff_thyer) a shout out for pointing me towards Pcapy. At one point, I was having some difficulty inspecting the 802.11 frames at a deep enough level using raw sockets. Pcapy allowed me to gain deeper insight into the 802.11 frames and ultimately fix some of the issues I was having. Thanks Joff!

## Resources

Here are some other resources that helped along the way.

- http://hackoftheday.securitytube.net/2013/04/wi-fi-ssid-sniffer-in-12-lines-of.html
- https://code.google.com/p/py80211/source/browse/Parse80211.py
- http://stackoverflow.com/questions/16801270/parsing-wifi-packets-libpcap

## Update #1

I've had a few folks report an issue with WUDS where the sensor is picking up probes that have a 0 RSSI, MAC addresses with a sequential first octet, and a garbled SSID. One of the users reported that commenting out the following line in the `/etc/network/interfaces` file fixed the issue on the Raspberry Pi.

```
pre-up wpa_supplicant -B -i wlan0 -c
/etc/wpa_supplicant/wpa_supplicant.conf
```

Apparently, `wpa_supplicant` is putting the wireless interface in a client state that is causing the issue. Killing the `wpa_supplicant` process fixes the issue temporarily. A permanent fix would be to make the above edit.

Like what you see? Join me for live training! See the Training page for more information.

---