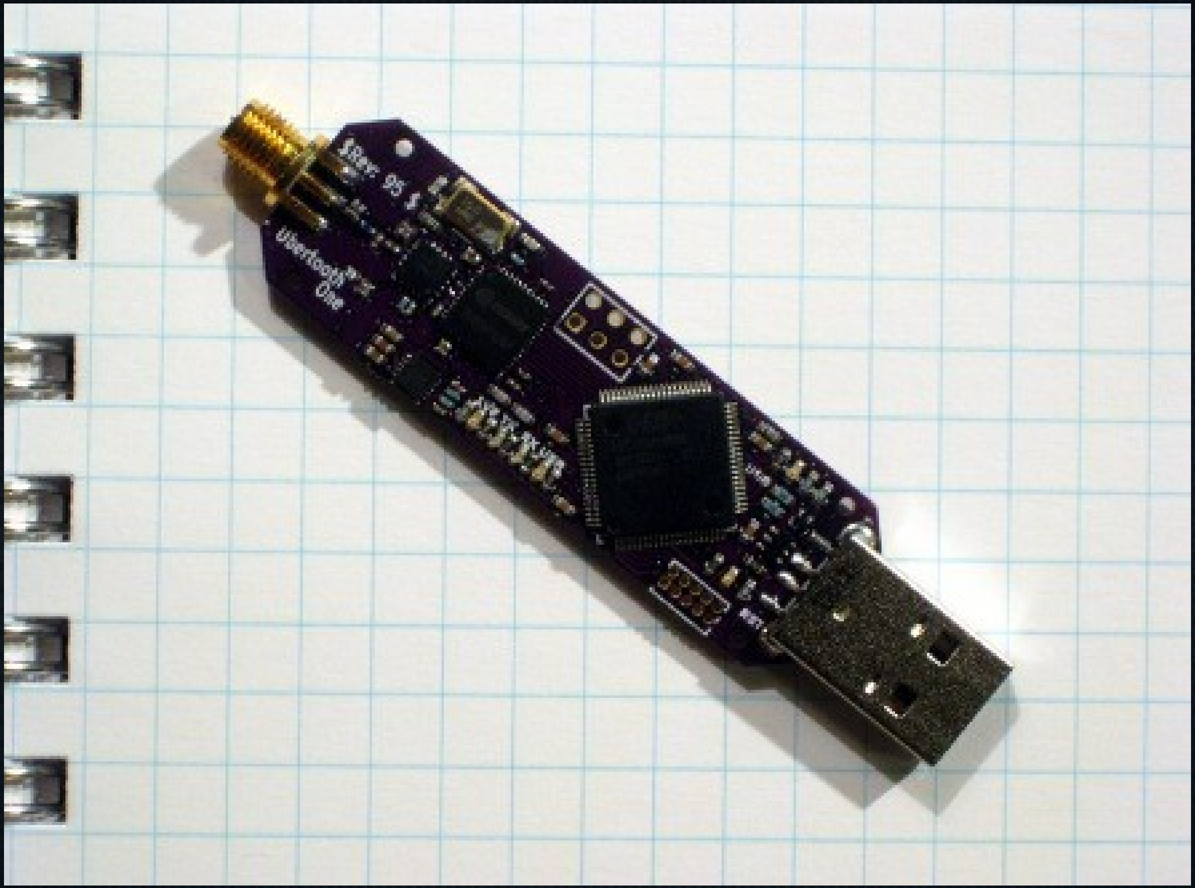
Bluetooth sniffing with Ubertooth



Dominic Spill
dominicgs@gmail.com

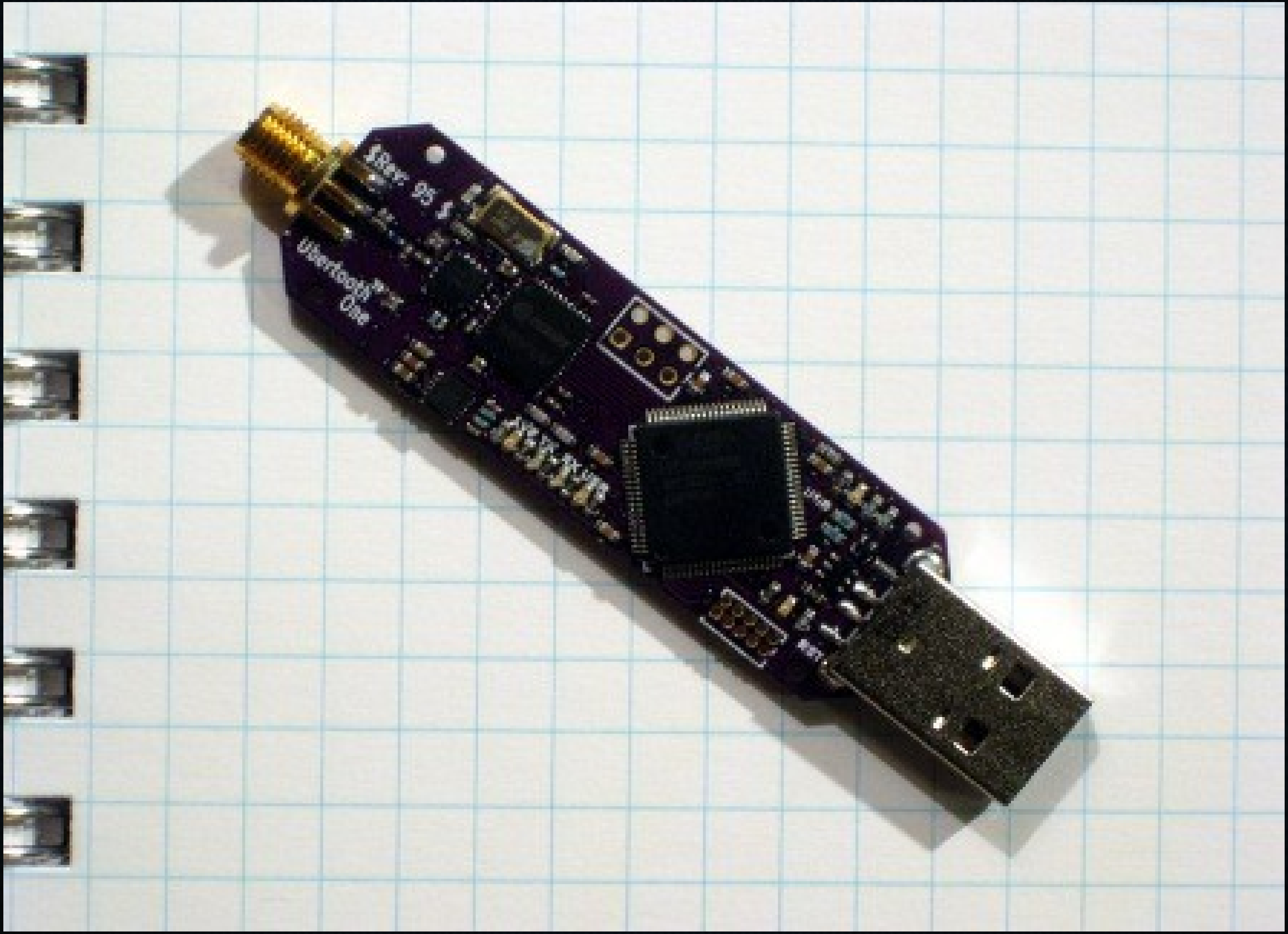
Dominic Spill

- Bluesniff: Eve meets Alice and Bluetooth
  - Usenix WOOT 07
- Building a Bluetooth monitor
  - Shmoo/Defcon/Toorcamp 09
  - With Michael Ossmann
- Work on Ubertooth / Daisho
- Life goal: sniff all the things

Warning

- If you wish to remain anonymous:
  - Remove your name from Bluetooth device names
  - Or turn off Bluetooth devices now
- Live demos at a con may not work
  - Especially when using 2.4GHz
  - This applies x2 at DEF CON

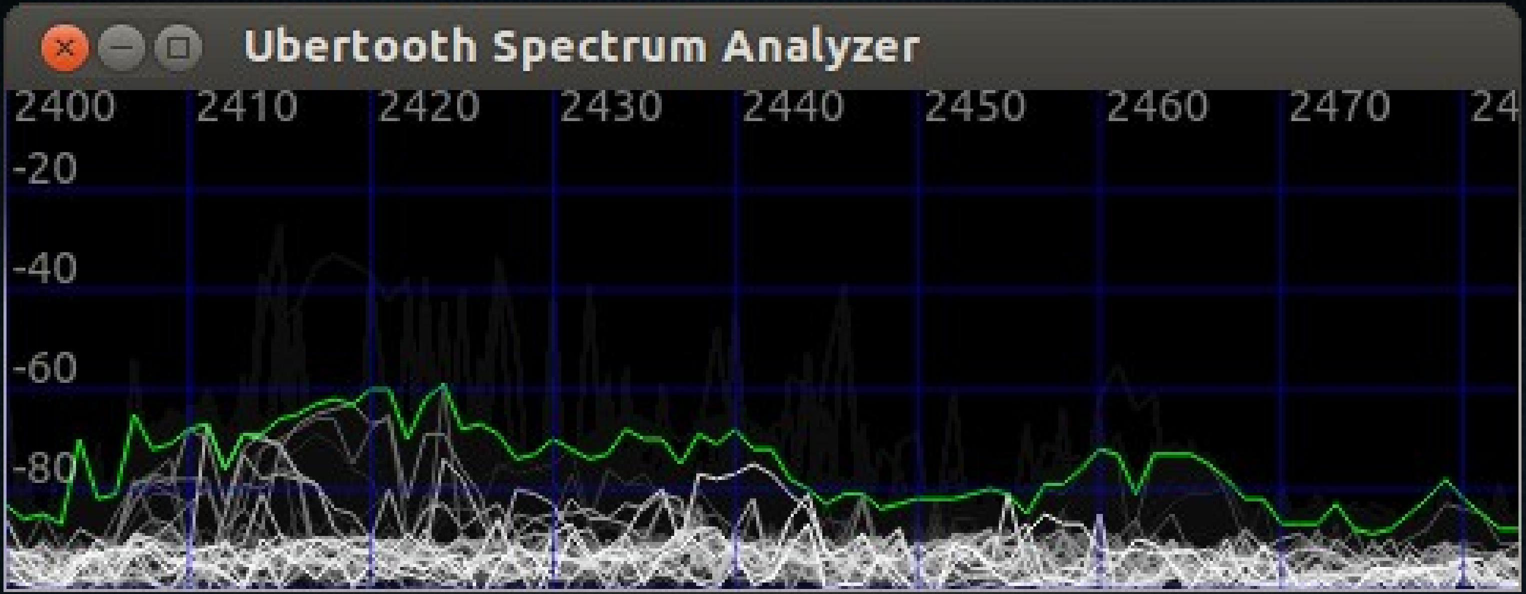
Ubertooth

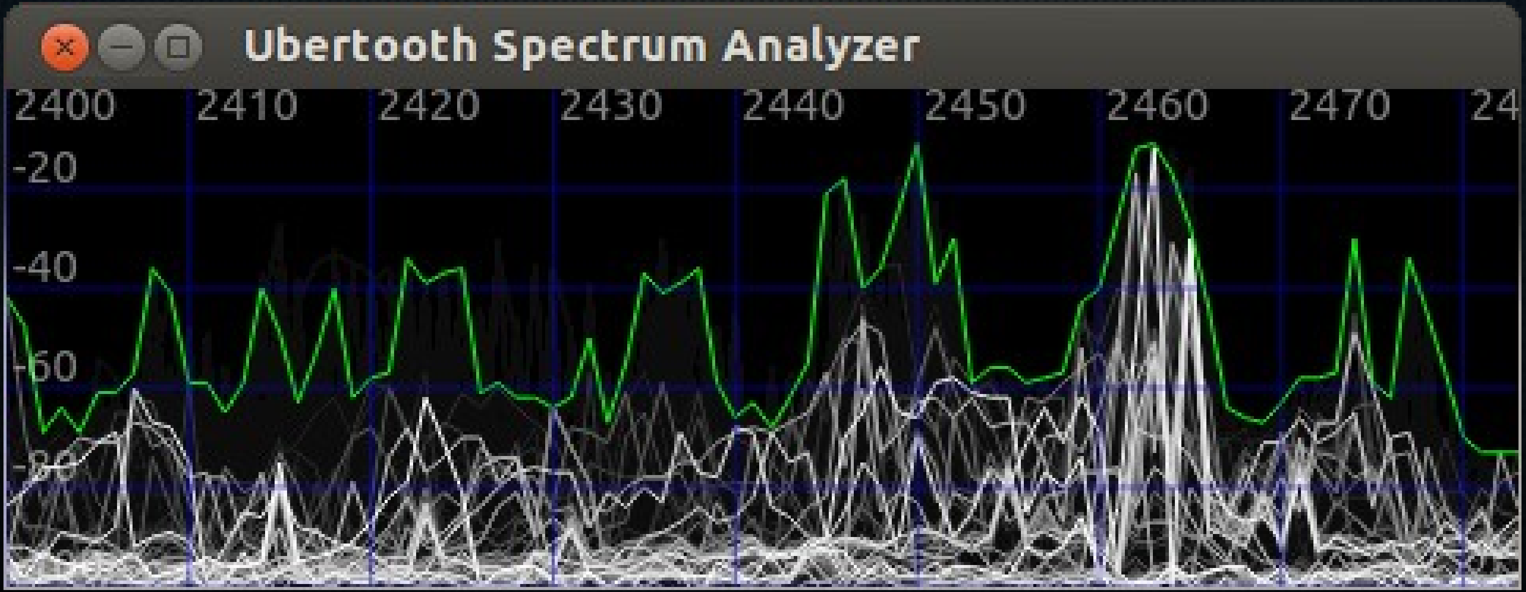


Ubertooth

- Designed by Michael Ossmann
- 2.4GHz experimentation platform
- Bluetooth 1.x, Low energy, 802.11 FHSS
- Hardware
  - CC2400 (+CC2591 frontend)
  - NXP LPC1756
  - USB device
- Open source software and hardware
  - http://ubertooth.sourceforge.net

Spot the difference?





Bluetooth

Bluetooth

- 2.4GHz ISM band
- Variable data rates
  - Basic Rate – 1Mb/s
  - Enhanced Data Rate – 3Mb/s
  - High Speed - Alternate MAC/PHY – 24Mb/s
  - LE (Smart) – 200Kb/s
- FHSS @ 1600Hz
  - 79 channels

Bluetooth

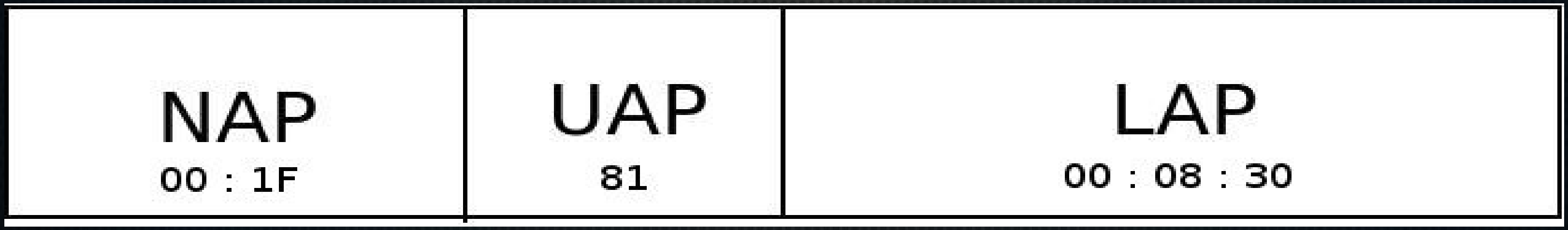
- Bluetooth SIG
  - 17,000 members
  - Free to join
- Bluetooth devices
  - 7 billion devices sold to end 2011
  - Will ship 2 billion devices this year
  - 20 billion expected in use by 2017

http://www.bluetooth.com/Pages/sig-membership.aspx

Bluetooth Sniffing is Hard

Bluetooth - Terminology

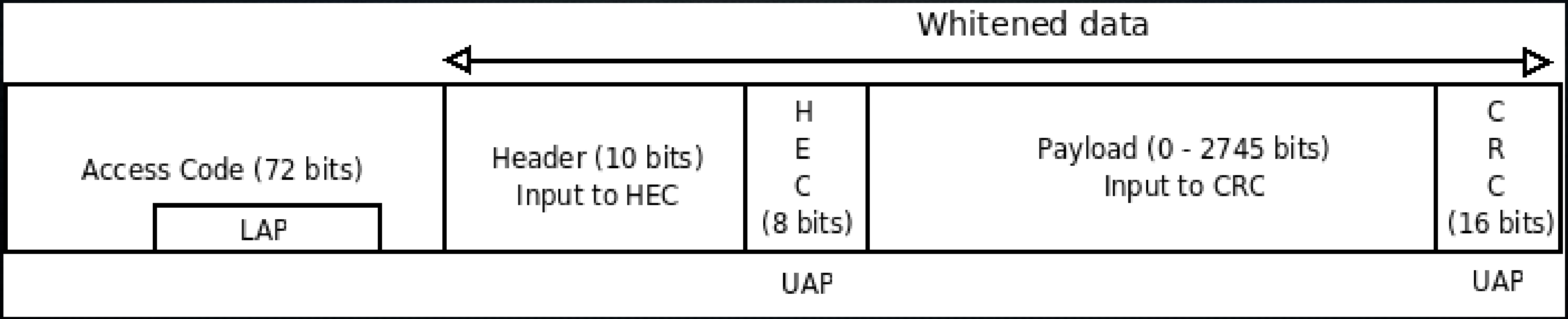
- Bluetooth device address / MAC
  - Three parts, not all present in packets
    - LAP - Lower – lowest 24 bits
    - UAP - Upper – next 8 bits
    - NAP - Non-significant – top 16 bits



- CLKN
  - 27bit 3200Hz internal clock
  - Increments twice per time slot

Bluetooth - Terminology

- Access code
  - Derived from LAP
- Packet Header
  - Error check based on UAP
- Payload
  - Possibly encrypted
  - CRC also based on UAP



Bluetooth - Terminology

- Non-Discoverable mode
  - Does not respond to inquiry scans
  - Still responds to page scans
  - Some newer devices ignore unknown page scans
- Data whitening
  - Packets XOR'd with pseudo-random sequence

Bluetooth sniffing is hard

- No “monitor mode”
  - Fixed correlator – not promiscuous
- Frequency hopping
  - 1600 hops/s
  - 625us/packet
  - Pattern based on MAC and CLKN
- Data whitening
  - PRNG initialised with CLK1-6
- Adaptive Frequency Hopping

Bluetooth sniffing is profitable (apparently)

- Known connection LE only - $250
- Known connection BR only - $10,000
- All channel BR/EDR/LE - $25,000

Packet Sniffing

Packets!

```
000010111110101110101010101010101010101010101010101010
101010101010101011011111001100001011011001100110011111
100110100101100011110010101011111111100000011100000000
000000000000011111111100000011110000111100001111000011
110000111100001111000011110000111100001111000011110000
111100001111000011110000111100001111000011110000111100
001111000011110000111100001111000011110000111100001111
000011110000111100001111000000001000011111000101010011
100011101101110011101101001101000100001000101010
```

Finding Packets – Old method

- Find access code
  - Treat 64bit chunks as possible access codes
  - LAP stored in bits 34-57
- Check access code
  - Check trailer (2 errors)
  - Generate access code from LAP
  - Compare access code to 64bit chunk (6 errors)

Flaws

- Slow on desktop CPU
- Unworkable on low power devices
- No errors allowed in LAP
- No error correction

Error Correction

Error Correction

- (64, 30) expurgated block code
  - Based on BCH (63, 30) code
  - Calculate syndromes to find error vectors

- Supposed to correct up to 6 bit errors
  - Too many false positive results
  - In practice correct <4 bit errors

Error Correction

- (64, 30) expurgated block code
  - Based on BCH (63, 30) code
  - Calculate syndromes to find error vectors

MATHS

- Supposed to correct up to 6 bit errors
  - Too many false positive results
  - In practice correct <4 bit errors

Error Correction

- Manufacturers don't implement it
  - Known access code loaded into correlator
  - Compared to received bits
  - Up to 6 bit errors
- This is what we do for a known address

Finding Packets – New Method

- Pre-calculate syndromes for n-bit errors
  - Use known access code
  - XOR with all possible n-bit error vectors
  - Generate syndrome for each error
  - Store in hash (uthash rules!)
- For each 64bit block
  - Calculate syndrome
  - Check hash for error vector
  - Correct error

Promiscuous Sniffing

- On a single channel
  - Sniff all packets
  - Correct up to 4 bit errors
  - Identify piconets
- Using multiple channels
  - Retrieve UAP from checksums
- Cannot retrieve NAP
  - That's ok, it's “non-significant”

Ubertooth-scan

- Finding non-discoverable devices
- Wright's Law
  - Security will not get better until tools for practical exploration of the attack surface are made available.

Frequency Hopping

Frequency Hopping – Local

- Ubertooth-follow
  - Follow a local Bluetooth device
  - Use bluez to extract CLKN
  - Push to Ubertooth
  - Start hopping

- Demo

Frequency Hopping – Local

- Pros
  - Reliable
  - Potentially sniff pairing
- Cons
  - Requires local BT device
  - No AFH support
    - Expected soon
  - Clock drift causes problems
    - This is fixable

Frequency Hopping – Remote

- Derive CLKN from received packets
  - Calculate hopping pattern for known address
  - Sniff single channel or hop randomly
  - Observe packets, timing and channel
  - Place packets in hopping pattern
  - Yields unique CLKN
- Calculate clock offset from CLKN → Ubertooth
- Send to Ubertooth
- Follow hopping piconet

Frequency Hopping – Remote

- Ubertooth-hop
  - Follow a remote piconet
  - Given LAP and UAP
  - Finds clock offset and hops
- Demo

Adaptive Frequency Hopping

- Part of Bluetooth 2.0 spec
  - Avoids noisy channels
  - Mostly avoids device's own wifi channel

- Demo
  - Very experimental

Kismet Plugin

- Plugin for current and upcoming Kismet
  - Only survey mode – static or sweep

- Demo

Recent Changes

PCAP / Wireshark Plugins

- Dissectors moving to Wireshark
  - Cleaned up
  - Will be built in to 1.10.? (ish?)
- New format
  - Libpcap linktype
  - PPI
  - Tool to convert

Bluetooth Baseband (libbtbb)

- Fixed up API
  - For other projects
  - GR-bluetooth (HackRF!)
-
- Thanks to “Will Code”

Bluetooth Smart

- AKA
  - Bluetooth Low Energy
  - Bluetooth 4.0
  - Wibree
- Much simpler protocol
- Mike Ryan presented at Toor/Shmoo/Black Hat
  - Sniffing
  - Injection
  - PIN cracking (crackle)

Platforms

- BeagleBone (Black)
  - Plenty of power
- Wifi Pineapple
  - Opkg packages
- Packages
  - rpm
  - deb
  - opkg

Future Work

- Adaptive Frequency Hopping
- Encryption / Pairing
- Transmit – packet injection
- Full LE stack
- Follow in Kismet
- Storage

Thanks to…

- Michael Ossmann
- Jared Boone
- Mike Kershaw (dragorn)
- “Will Code”
- Mike Ryan
- Zero Chaos

Questions?

dominicgs@gmail.com
Twitter: dominicgs
Slides: dominicspill.com/defcon/slides2013.pdf