Api-do: Tools for Exploring the Wireless Attack Surface in Smart Meters

Travis Goodspeed
University of Pennsylvania
Philadelphia, PA
twitter.com/travisgoodspeed

Sergey Bratus, Ricky Melgares, Ryan Speers, Sean W. Smith
Dartmouth College
Hanover, NH
sergey,melgares,rspeers,sws@cs.dartmouth.edu

***Abstract*—Security will be critical for the wireless interface offered by soon-to-be-ubiquitous smart meters — since if not secure, this technology will provide an remotely accessible attack surface distributed throughout many homes and businesses. However, history shows that new network interfaces remained brittle and vulnerable (although believed otherwise) until security researchers could thoroughly explore their attack surface. Unfortunately, for the majority of currently available smart meter wireless networking solutions, we are still in that pre-exploration phase; "closed" radio stacks with proprietary features impede exploration by posing multiple hardware and software obstacles to standard network attack surface exploration techniques. In this paper, we address this problem by presenting open and extensible software tools for 802.15.4-based proprietary stacks that work on commodity digital radio platforms. We hope this contribution advances the state of the art beyond the pre-exploration tipping point, and toward real security.**

***Keywords*-802.15.4, ZigBee, attack surface exploration**

I. Introduction

Soon-to-be ubiquitous smart meters[1] provide wireless network interfaces that are new and unexplored. This combination is worrisome. Wireless network interfaces present a potentially significant security risk, since they potentially expose internal systems and networks to any attacker within radio range—which generally spills beyond typical physical perimeters such as locked doors and tall fences. Furthermore, history suggests that new kinds of network interfaces, although initially believed secure by some, too often turn out to be brittle until tools exist for the security research community to explore their attack surfaces.

Our tools, collectively named *Api-do*, as a pun on apiculture (the craft of beekeeping) and the defensive martial art of Aiki-do, are to promote the culture of defensive attack surface exploration for 802.15.4, building on Joshua Wright's KillerBee [36].

Looking Back: 802.11: For example, let us look back to the rollout of a previous widespread wireless network interface, 802.11. The initial deployments were replete with significant security holes that were identified and addressed by defenders once open tools existed to explore the attack surface.

We enumerate a few examples:

- Initially, vast numbers of office WiFi networks were completely open to sniffing. Because RF monitor mode was hard to enable on commodity interfaces, the community overlooked such attacks. The emergence of open tools such as Kismet [23] and long-range sniffer demonstrations such as Shmoo Group's "WiFi Sniper Rifle" led to solutions.
- Initially, vast numbers of WiFi firmware and drivers were susceptible to attacks based on bit-by-bit crafted frames. Because commodity firmware made injecting crafted frames difficult, the community overlooked such attacks. The emergence of Mike Lynn's "Airjack" driver [33] led to solutions; custom open drivers for software-configured Atheros chips led to another wave of improvements.
- Initially, WiFi depended on the fundamentally flawed cryptography of WEP. Because of the difficulty of sniffing in commodity interfaces the community overlooked these problems; because of difficulty with injection in commodity interfaces, initial defensive WEP add-ons such as "weak" IV filtering and dynamic WEP keying proved inadequate as they did not account for traffic re-injection attacks. The emergence of KoreK's "chopchop,"[25] using access points as oracles for tweaked frames, led to solutions.
- Initially, WiFi clients did not authenticate access points, enabling man-in-the-middle attacks and information disclosure. Because of the limited support for infrastructure Master mode in commodity interfaces, the community overlooked this fundamentally flawed authentication model. The emergence of AP emulation tools such as "Karma" [22] led to solutions.

History thus shows that network attack surface exploration techniques are the fundamental building blocks that enable researchers to reason about threat models. Their use provides

[1]The Edison Foundation Institute for Electric Efficiency compiled a list of target deployments and found that over a 50 million meters will be in operation by 2019, which will represent almost half of nationwide households [8]. See also [19], [5].

researcher with the information required to prioritize and prune the branches of attack tree, based on finding out what is likely to be feasible attack, and what is not, and to estimate the amount of effort required of the attacker to proceed along a particular attack tree branch.

This information is crucial to building a realistic threat model, since the latter is built on assumptions about feasible attacker achievements, their likely cost-vs-benefit to the attacker, and in turn (combined with the expected attacker profile) informs the risk assessment.

Looking Forward: 802.15.4: To offset such scenarios in the smart meter wireless networking, we present open, commodity-based implementations of radio stacks which provide compatability with several closed radio stacks which add proprietary extensions atop of 802.15.4 to the extent sufficient to implement basic exploratory and offensive techniques such as *sniffing*, *geographic network mapping*, *crafted frame injection*, and *selective jamming*.

The tools we present here also helped us demostrate in [13] that for some digital radio protocols including 802.15.4, even a Faraday cage around an RF network is not a cure-all protection from remote attacks on its PHY and link layes. In particular, an attacker who does not own a radio but can control higher layer protocol payloads may in fact be able to inject crafted **PHY** frames into unencrypted digital radio links using our *Packet-in-Packet* technique.

We built our tools on top of and around Joshua Wright's KillerBee 802.15.4 exploration suite, which we significantly extended and adapted to several commodity digital radio platforms, such as the Tmote Sky[2] with the freely available GoodFET firmware and programmer functionality – a deliberate choice to make the resulting kit as affordable and accessible as possible.

The aforementioned techniques are not attacks per se; an adversary capable of successfully replicating them does not necessarily get the achievement they help characterize. Instead, they help delineate the major "bottlenecks" that the attacker must pass on the way to these achievements, and the payoff of attempting to pass such bottlenecks.

For example, whether an attacker may or may not be able to remotely exploit wireless link-layer drivers depends on how well firmware and drivers validate incoming traffic. In proprietary system where firmware and driver source code is not available, the likelihood of buggy validation can only be established by exploring the device's response to a broad range of sample inputs. In particular, it usually becomes apparent whether the device's network stack implementers understood the need for such validation, or merely assumed that only well-formed traffic from trusted sources will reach a particular layer of their stack. The answer radically affects the threat model.

This Paper: Section II presents our tools for sniffing and geolocation. Section III presents our tools for frame crafting and injection. Section IV presents our exploration of jamming. Section V concludes.

II. Network Discovery

To analyze the security of wireless networks, one first needs to be able to discover them, either passively or actively.

A. Multi-channel monitoring a must

Unlike 802.11, where discovery can be easily accomplished by monitor mode sniffing, this step can be non-trivial for other digital radio networks. For example, NordicRF network links can only be sniffed if the sniffer knows the MAC address of the targeted node, and a similar situation exists in Bluetooth links [17].

This makes promiscuous sniffing difficult both by configuration and by hardware, and makes identification of the available networks necessary for reliably capturing their traffic, not vice versa, and makes it necessary to use more sophisticated techniques for such identification – across all available channels. The necessity for convenient tools to do so is also recognized by other researchers, in particular Kevin Finisterre [11].

Moreover, even having captured some 802.15.4 frames on a given channel, one should not assume that the relevant network traffic is limited to that channel (as would be the case in common 802.11 or 802.15.4 deployments). Instead, the protocol may be using multiple channels at once, in intricate patterns.

We therefore developed *OpenEar*, an all-channel monitor application for integration with the KillerBee framework, in order to enable such multi-channel sniffing in a wardrive-like enviroment, including capturing fast-frequency-hopping devices with unknown patterns.

The Wireless Energy Management System experience.: Our first encounter with the need for an all-channel monitor occurred when attempting to survey the operation of the wireless energy management system (WEMS) in use by Dartmouth College's Facilities Operations and Management, as part of the campus energy management system. This is a commercial product which has been widely accepted and deployed across the commercial and institutional communities. It integrates with such functions as automatic billing, among other features.

When attempting to capture traffic from the WEMS network, we noticed that the same PAN ID[3] of the target network would appear on different channels, which is not

[2]The Tmote Sky by Sentilla is a low cost, low power wireless sensor module that is IEEE 802.15.4 compliant. It utilizes a Texas Instruments MSP430 microcontroller and contains an array of sensors on board, as well as a Chipcon 2.4GHz IEEE 802.15.4 wireless transceiver that is controlled by the MSP430 through an SPI port, all in an integrated USB dongle package capable of running off two AA batteries, making it a highly convenient and portable platform for field-use.

[3]The Extended PAN ID is a 64-bit identifier, which is shortened a 16-bit PAN ID for use in most packets.

standard 802.15.4 behavior, since the 802.15.4 standard specifies that a chosen PAN ID must be unique from those of neighboring networks in order to avoid PAN ID conflicts. Further explanation showed that the system was using a *proprietary channel-hopping algorithm* that continuously determines the 4 best channels at any given time from the available 16 IEEE 802.15.4 channels, and dynamically switches from this group of four channels[4], for reliability in real-world environments and interference avoidance in overlapping Bluetooth and 802.11 channels, since they share part of the 2.4 GHz spectrum.

B. Tools to Enable Network Discovery

OpenEar: is a 16-channel monitor we built on the KillerBee framework by assigning each attached device a unique channel to listen on, so that 16 devices may be used for simultaneous monitoring of all channels.

OpenEar also extends the KillerBee framework with user-space threading of the capture process, so that a single program is able to initialize and shut down capture processes for all channels at once. This differs from previous work by security researchers such as Kevin Finisterre [11], since his method of building an all-channel monitor involved scripting the start-up of a single non-threaded process for each device and channel using a Bash shell script.

Threading an all-channel capture process also enables the sharing of resources between capture processes, such as location information from an attached GPS device (see II-C).

The OpenEar application enumerates all of the attached devices, assigns a channel number to each, initializes a packet capture file for each channel and, optionally, a database connection for logging information to a MySQL server using a custom database schema we designed for the purpose.

OpenEar includes a persistent in-terminal display for displaying real-time status, which is an important feature for performing real-time analysis and identifying "chatty" or saturated areas of service when wardriving or on-site (cf. *Kismet* for 802.11, which displays a continually updated digest of wireless network information gleaned from traffic).

zbWarDrive: is another tool we created in order to efficiently allocate and use a limited number of devices for capturing traffic at any given time in an area, where a small number of networks may be present and where frequent channel-hopping is not expected. This tool makes use of injection to inject a beacon request frame and listen for the beacon response from the network. This expected response is used to determine if a network is operating on this channel, and if so, a capture device is assigned to monitor it.

[4]This group of four channels changes frequently as the system adapts to changing enviromental RF interference conditions.

C. Location Logging and Estimation

Although KillerBee provides a location estimation tool, this tool acts as an RSSI strength meter, which in our experience proved to be an inefficient method for locating devices during site surveying and wardriving. We therefore extended the KillerBee framework by adding the ability to capture and log location information along with captured packets.

To enable the logging of location information, we used `gpsd` [18], an open-source daemon that accepts data received from an attached GPS device, formats the GPS data into a JSON format, and returns the GPS data over a local TCP/IP connection.[5]

In order to tag packets with location information, the OpenEar application initializes a "location" thread in addition to the capture threads. The sole purpose of the "location" thread is to continuously poll the `gpsd` daemon service and update the current location (latitude, longitude, altitude), referenced each time a packet is logged. This location data is fed into the database (if connected) and can be logged into PCAP files using the PPI-GEOLOCATION specification [10], which uses per-packet information tags. Since Wireshark, Scapy, and Kismet currently support the PPI-GEOLOCATION specification, we find this standard to be a natural choice for logging location information across a variety of capture applications and storing it along with packets.

Logging location information is important for wardrive post-analysis and geolocation. Location information can be used to identify saturated (or weak) areas of service or, with sufficient data, to pinpoint the exact location of a device or network in order to profile the network or to carry out physical attacks against the hardware, such as the extraction of encryption keys from the onboard hardware, as has been done with ZigBee and 802.15.4 radio chips in the past [15], [14]. Figure 1 shows a plot of 802.15.4 traffic across the Dartmouth campus captured with OpenEar.

A more advanced geospatial analysis takes into consideration the received signal strength and grouping of packets by network or relationship. For the purposes of such analysis, we experimented with the use of GIS applications such as *ArcMap* [4], part of Esri's ArcGIS suite of geospatial processing programs, which provide the ability to perform advanced spatial interpolation. We used the hotspot analysis interpolation tool in ArcMap to interpolate and estimate wireless coverage across the campus, based on a limited number of sample points and strength readings. Figure 2 shows a result of such interpolation.

[5]`Gpsd` allows multiple applications to share and use a GPS device concurrently. This is necessary as most GPS devices capable of sending GPS data to a computer do so over a serial connection, which cannot be shared or accessed by multiple applications at once. `Gpsd` provides a Python API, which facilitates the integration of `gpsd` into the KillerBee framework, which is also written in Python.

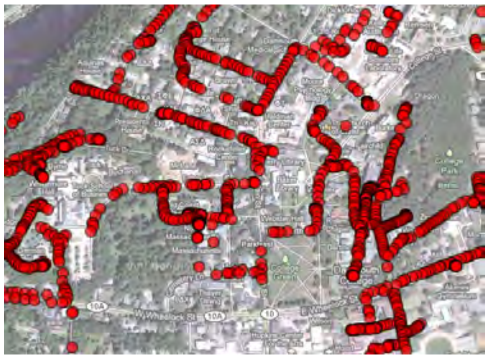

Figure 1. 802.15.4 packets captured on Dartmouth campus. Each circle represents a point where an 802.15.4 packet was logged.

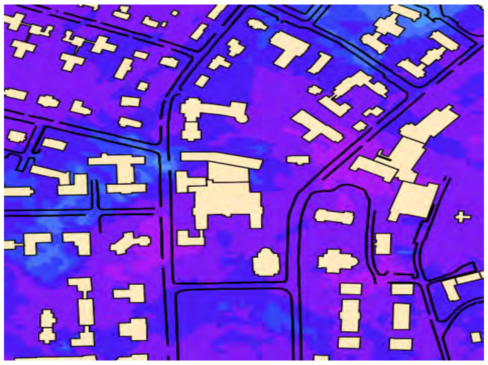

Figure 2. ArcMap spatial interpolation using the hotspot analysis tool (intensity proportional to signal strength)

In order to be able to import data into GIS applications such as ArcMap, we have developed scripts for extracting location data from the database and converting into the shapefile format, a geospatial vector data format accepted by these applications. Our `csvgps2shp.py` [30] utility takes in a csv file with records of the form "ID, RSSI, latitude, longitude", where ID is either the name of the network or device, and RSSI is the received signal strength, and creates a shapefile using the open-source GDAL (Geospatial Data Abstraction Library) framework, which includes Python bindings [12].

III. 802.15.4 Frame Crafting and Injection

The capability to craft packets and frames and inject them into the network is one of the most important tools for their practical exploration. Indeed, the progress of network testing tools was driven by libraries (such as *libnet*, *libdnet*, and the *Scapy* scripting framework for TCP/IP, and, more recently, *LORCON* for 802.11) driving this capability at the lower layers of the OSI stack model [31].

To facilitate frame construction, we implemented `dot15d4`, a layer extension for Scapy that implements a subset of the IEEE 802.15.4 protocol. We use it extensively for *inferenced packet generation* experiments, where injected frames and packets are adjusted to fit the network environment.

A. *Scapy `dot15d4` extension*

The infromation about the practical use of the 802.15.4 protocol was taken from a variety of sources, including the IEEE 802.15.4 Specification [20], [21], ZigBee Specification [37], and Daintree Network's Getting Started with ZigBee and IEEE 802.15.4 primer [7]. As of this writing, our tool supports crafting of all frame types, and of most subtypes and features, with the exception of GTS and pending addresses and some command subtypes that are not yet fully implemented. The current status of the implementation is shown in Table I.

For further information please refer to [29], [26].

Class Name	Notes
Dot15d4	
Dot15d4FCS	Includes checksum.
Dot15d4Ack	
Dot15d4Data	
Dot15d4Beacon	GTS & pending addresses not fully implemented.
Dot15d4Cmd	All subtypes are not yet implemented.
Dot15d4Cmd -CoordRealign	
Dot15d4Aux -SecurityHeader	

Table I
Implementation Status of Frame Types in Scapy `dot15d4`

The 802.15.4 beacon specification alone, not even including the ZigBee beacon specification, has a number of variable length list fields, fields whose values are dependent on other fields, and fields whose existence is dependent on other fields. One example is the frame control field source addressing mode, which determines whether the frame's source address field exists and its length. Figure III-A demonstrates the complexity of the structures of 802.15.4 beacon frames, as an example.

To address these issues, and make crafting of arbitrary frames simpler, our Scapy `dot15d4` layer[6] handles the relations between these interconnected fields for the subset of the 802.15.4 protocol that is implemented in Scapy. Although Scapy provides ConditionalField interfaces that can be adapted to work well for the questions of existence of fields based on other fields values, it is more difficult to force the value of a field based on the value of another field. In many cases, we made the design choice of leaving these dependencies untouched by Scapy, so as to allow security researchers to craft frames that do not adhere strictly to the

[6]Our `dot15d4` layer has been adopted as the base layer for the Cesar Bernardini's Scapy `6loWPAN` layer, and we hope the community will continue to be able to build upon this tool to enable further research.

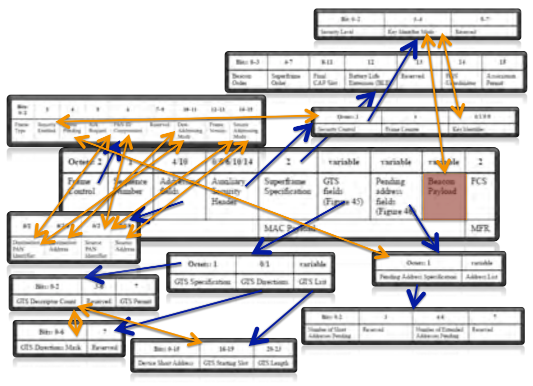

Figure 3. Relationships and dependencies between fields in 802.15.4 Beacon Frames

802.15.4 standard for applications such as intelligent fuzzing or fingerprinting.

B. *Inferenced packet generation*

We developed techniques for gathering data about networks from a combination of active and passive listening. This information is loaded into the central database over a "learning period." Data from multiple frames captured can be merged to inference data, which may not be available from any single frame. Using this data, tools can craft packets with the user having to define only minimal data. For example, given a type of frame and a target (destination) short address, the tool (a number of functions in `zbForge.py`) can pick an appropriate source address that was observed communicating with the target, and use that address in the packet being constructed. The tool will also use a sequence number that follows from the most recent number observed, with some randomization added. The goal of this tool is to require minimal user effort or interaction, yet produce realistic looking packets, which are more likely to be accepted by target devices and which are more difficult to detect in packet analysis than packets generated without inferencing.

The database is programmed with a number of views to quickly return the most recent "inferred" status between devices. The tool retrieves this data based on the parameters supplied to it in the `makeLinkData` function. The output of this function, together with the desired type of frame is used to generate a Scapy `dot15d4` object which can then be injected.

Figure 4 illustrates this process.

If a user specifies that they would like to spoof a beacon frame, they need only to specify the sending device, and the remaining information (such as the sequence number, network PAN ID, long address) will be inferred. Many other types of packets similarly need only minimal information

```
kb = killerbee.getKillerBee(channel)
link = makeLinkData(srcTarget, dstTarget)
_, scapy = create(link, FRAME_802_DATA)
print "Sending forged frame:"
scapy.show2()
kb.inject(str(scapy))
```

Figure 4. Example Usage of zbForge Functions

(usually sender and receiver) specified by the user, and the remainder of fields are calculated.

IV. Reflective and Selective Jamming

The capability to suppress network messages is an important complement to crafting and injecting them. The effectiveness of their various combinations for manipulating the state of network's nodes had been shown early on for attacks on TCP/IP (e.g., the "Mitnick attack" [27], Ptacek and Newsham's seminal NIDS evasion paper [28], etc.), and is pivotal for a variety of wireless attacks.

Accordingly, we explored message suppression in the wireless medium by using RF (radio frequency) jamming techniques to intercept and corrupt traffic.

Further, to enable several common exploration, exploitation, and attack techniques, we need to be able to reflexively and selectively jam certain transmissions. We implemented reflexive jamming techniques to avoid some jamming defense strategies which try to differentiate jamming (intentional denial-of-service) from other non-malicious issues in the network. Wood claims that "a node can easily distinguish jamming from the failure of its neighbors by determining that constant energy, not lack of response, impedes communication" [35, 50]. However, a reflexive or selective jamming attack, as implemented here, have no constant energy, and even most devices listening in a monitor/promiscuous mode will not see the jamming frame independently of the frame that it jammed. Many authors classify such an attack as a collision attack, to separate it from 'traditional' jamming. Wood states that these malicious collisions "create a kind of link-layer jamming, but no completely effective defense is known" [35, 51].

A. *Reflexive Jamming on the Tmote Sky with GoodFET*

We implemented reflexive jamming on the Tmote Sky device using GoodFET firmware as a basis, and our functionality been added to the public GoodFET CCSPI firmware (as verb `0xA0`).

The `goodfet.ccspi` application first places the radio into promiscuous mode, disables CRC checking, and tunes to the desired channel before placing the firmware into reflexive jamming mode. Reflexive jamming has been tested against communications and seems to have constant effectiveness against packets greater than 12 bytes in length (due to the inevitable delay between registering a frame event and preparing the radio for the jamming transmission).

Corruption typically occurs between the 8th and 12th byte of the frame. The jam is triggered by the SFD (Start of Frame Deliminator) line on the Chipcon2420 radio going high, as shown in Figure 5. When the firmware application observes this condition, it immediately shifts the radio into TX mode, loads a frame into the TXFIFO buffer, and sends the STXON strobe command to start transmission without the radio performing the CCA (clear channel assessment) which is designed to prevent collisions.

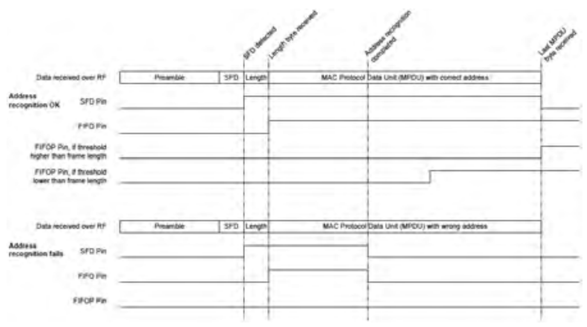
SFD detected
Length byte received
Address recognition completed
Last MPDU byte received
Data received over RF
Preamble
SFD
Length
MAC Protocol Data Unit (MPDU) with correct address
Address recognition OK
SFD Pin
FIFO Pin
FIFOP Pin, if threshold higher than frame length
FIFOP Pin, if threshold lower than frame length
Data received over RF
Preamble
SFD
Length
MAC Protocol Data Unit (MPDU) with wrong address
Address recognition fails
SFD Pin
FIFO Pin
FIFOP Pin

Figure 5. Chipcon 2420 Receive Operation Pin Values [32, 34]

Figure 6 shows a series of beacon frames, with frame 13 and 14 being sent when jamming was not enabled, and frames 15 and 16 being sent once reflexive jamming was enabled. Note that the eighth byte (`0x46`) in the frame is the first byte to be corrupted by the reflexive jam. This means that in the current implementation, frames of 7 bytes or shorter cannot be reflexively jammed, however, most frames of interest, including virtually any frame with a payload, will be of sufficient length to jam with this technique. Acknowledgement frames are not currently vulnerable, but the Chipcon2430 chip, which has the radio and an 8051 MCU on the same die, may enable successful jamming on such shorter frames in future work.

If this attack is done without acknowledgment spoofing, the sending device may repeatedly try to transmit its packet as it is not being acknowledged. This means that an exhaustion attack on a sending device can be furthered using this technique. The sending device would not know where in the frame the collision occurred in the frame, as it cannot monitor while it is transmitting due to the basic properties of radio media. An intrusion detection system may be able to monitor frames if it knew the expected frames to determine if collisions were occurring later in the frame, which *may be* statistically unusual in non-malicious interference.

B. Acknowledgment Spoofing

In an acknowledgment spoofing attack, we want to prevent some frames from being received at their destination, while making the sender believe that these have been received. In the IEEE 802.15.4 specification, the acknowledgment bit is set in the frame control field if a sender desires their transmission to be acknowledged by the receiver.

```
Frame 13: 13 bytes on wire, 13 bytes captured
IEEE 802.15.4 Beacon, Src: 0xef01
  FCS: 0x9fba (Correct)
0000  00 80 ac 01 39 01 ef 46 cf 00 00 ba 9f

Frame 14: 13 bytes on wire, 13 bytes captured
IEEE 802.15.4 Beacon, Src: 0xef01
  FCS: 0xd247 (Correct)
0000  00 80 ad 01 39 01 ef 46 cf 00 00 47 d2

Frame 15: 13 bytes on wire, 13 bytes captured
IEEE 802.15.4 Beacon, Src: 0xef01, Bad FCS
  FCS: 0x3b7c (Incorrect, expected FCS=0x215a
  [Expert Info (Warn/Checksum): Bad FCS]
[Malformed Packet: IEEE 802.15.4]
0000  00 80 ae 01 39 01 ef 93 99 99 b9 7c 3b

Frame 16: 13 bytes on wire, 13 bytes captured
IEEE 802.15.4 Beacon, Src: 0xef01, Bad FCS
  FCS: 0x3a42 (Incorrect, expected FCS=0xcbea
  [Expert Info (Warn/Checksum): Bad FCS]
[Malformed Packet: IEEE 802.15.4]
0000  00 80 af 01 39 01 ef 33 33 33 33 42 3a
```

Figure 6. 802.15.4 Beacon Frames Undergoing Reflexive Jamming

The acknowledgment frame type simply consists of a 2 byte frame control field (FCF), 1 byte sequence number (matching the sequence number of the frame being acknowledged), and a 2 byte frame check sequence (FCS). The FCF is known in advance for the acknowledgment field, and the FCS is a non-cryptographic checksum that can be calculated prior to sending the packet. The one piece of data which needs to be captured from the original packet is the sequence number. The sequence number is always the third byte of a 802.15.4 frame.

Most collision attacks need a frame to be generated and transmitted after another one is jammed, and these depend on the specific style of the attack. Implementation can be done using Scapy `dot15d4` and KillerBee, or even using the `zbForge` library to assist in packet generation, and then the frame can be transmitted either by the reflexive jamming device or another radio interface. Choosing the proper frames to jam, and generating the proper preceding or follow-up frames is key to making an effective, as well as targeted, attack.

C. Selective Jamming on the USRP2

Selective jamming is the process of listening for a specific transmission or packet, and disrupting it by transmitting noise or a packet at the same time. Although this technique has been covered numerous times by academic researchers in the past [24], the aim is to lower the barrier of entry by implementing the selective jamming technique on a platform that is commonly used by security researchers: the Ettus

Research USRP2 [9].[7]

To implement selective jamming on the USRP2, we ported the UCLA framework [6] to the USRP2 and implemented traffic recognizer and jammer/transmitter signal processing blocks, while using the existing receiver and demodulation components provided by the framework to enable the reception of 802.15.4 traffic. Figure 7 shows the selective jamming architecture.

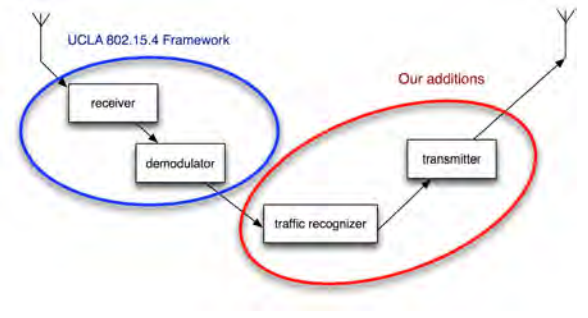

Figure 7. Selective Jamming Architecture on USRP2

Given a bit mask to match against, the traffic recognizer reads the incoming packet byte by byte, comparing the bytes against the mask. Upon a match, the transmitter is activated, which simply generates a burst of noise on the given channel, although a random packet could be sent as well.

One limitation encountered while working with the USRP2 was the inability to easily analyze packets in real time so that the transmission can be disrupted before it finishes. For all of the USRP2's glory, the latency between the USRP and the host computer prevents one from jamming packets in real-time or within the same frame; the analysis logic cannot therefore reside on the host computer. One solution is to implement the traffic recognizer component directly on the FPGA, which requires modifying the firmware on the FPGA, written in VHDL. A recently released paper on reactive jamming in regards to 802.15.4 and the USRP does exactly just this [34]. The required effort, however, is comparable with implementing the technique on an much cheaper microcontroller, such as the Tmote Sky with GoodFET.

Despite this limitation, the USRP2 was able to inference and jam particularly long sequences or exchanges, such as that which occurs when a device joins a network. Similarly, if the ACK bit for a particular packet is set, this will indicate that an ACK is expected as the next packet, allowing us to inference and activate our jammer ahead of time to selectively jam the ACK.

[7]We note that this is the only non-commodity component of our tool suite, as the rest of them use commodity Chipcon-based and comparable digital radio platforms. The commodity chipset devices on the market at the time of research were not able to reliably execute this attack, however we expect to rectify this problem in future work.

V. CONCLUSION

We conclude with some thoughts on commodity hardware vs software-defined radio.

In our effort to get the 802.15.4 exploration tools in the hands of as many researchers and asset owners as we can reach, we purposefully avoided the academically popular software-defined radio platforms whenever possible. Our vision of a proper, broadly available toolkit is a combination of a laptop-hosted software and an affordable USB stick peripheral.

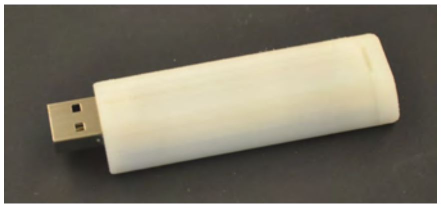

Figure 8. Our Custom Field-Ready Casing on an Atmel RZUSBSTICK

Unfortunately, software-defined radio peripherals such as USRPs are not only expensive but also require considerable expert knowledge to operate. The contrast with commodity radio chips is striking, in that commodity digital radio chip-based platforms would facilitate easy access for asset owners, engineers, and even vendors' own testers, increasing the size of the smart meter wireless research community possibly by an order of magnitude if not more.

In conclusion, we would like to quote Joshua Wright, a pioneer of practical wireless security, whose message we fully support:

> *Practical security does not improve until tools for exploration of the attack surface are made available.*

ACKNOWLEDGMENTS

The authors thank Joshua Wright, Michael Ossmann, and Kevin Finisterre for lending their expertise and support. This research was supported in part by the Department of Energy under Prime Award No. US DOE DE-OE0000097, Subaward No. 2010-01251-01 and in part by the National Science Foundation under Grant Award Number 1016782. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the DOE or NSF.

REFERENCES

[1] Aircrack-ng. http://www.aircrack-ng.org/.

[2] Airbase-ng. http://www.aircrack-ng.org.

[3] Airpwn. http://airpwn.sourceforge.net/Airpwn.html.

[4] Esri. "ArcGIS." http://www.esri.com/software/arcgis/.

[5] Chaudhary, R. (2010, November 17). In-State: Smart Energy to Fuel 802.15.4 and ZigBee Adoption. Smartgrid TMCNET.

[6] Choong, Leslie, "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio," UCLA, 2009.

[7] Daintree Networks. "Getting Started with ZigBee and IEEE 802.15.4." www.daintree.net/downloads/whitepapers/zigbee_primer.pdf. 2008.

[8] The Edison Foundation Institue for Electric Efficiency, "Utility-Scale Smart Meter Deployments, Plans and Proposals - February 2010." http://www.edisonfoundation.net/iee/issuebriefs/IEE_SmartMeterRollouts_update.pdf.

[9] Ettus Research. "USRP2." http://www.ettus.com.

[10] Ellch, Jon. "Per-Packet Information specification for Geolocation." Version 1.2.0. Crucial Security, Inc., March 2011. http://new.11mercenary.net/~johnycsh/ppi_geolocation_spec/.

[11] Finisterre, Kevin. "Successful Zigbee Wardrive Rig is Online!" Joint Direct Attack Munition Smart Weaponry Blog. http://www.digitalmunition.com/_/Blog/Entries/2010/10/13_Successful_Zigbee_Wardrives!.html. October 2010.

[12] Open Source Geospatial Foundation. "Geospatial Data Abstraction Library." http://www.gdal.org.

[13] Goodspeed, T.; Bratus, S.; Melgares, R; Speers, R.; Shapiro, R. Packets in Packets: Orson Welles In-Band Signaling Attacks for Modern Radios. 20th USENIX WOOT. 2011.

[14] Goodspeed, Travis. "Extracting Keys from SoC Zigbee Chips." http://travisgoodspeed.blogspot.com/2009/08/extracting-keys-from-soc-zigbee-chips.html.

[15] Goodspeed, Travis. "Breaking 802.15.4 AES128 by Syringe." http://travisgoodspeed.blogspot.com/2009/03/breaking-802154-aes128-by-syringe.html.

[16] Goodspeed, Travis. "GoodFET Project." http://goodfet.sourceforge.net.

[17] Goodspeed, Travis. "Promiscuity is NRF24L01's Duty." http://travisgoodspeed.blogspot.com/2011/02/promiscuity-is-nrf24l01s-duty.html.

[18] "gpsd." http://gpsd.berlios.de.

[19] Itron. "Itron Selected by CenterPoint Energy as AMI Technology Provider: Agreement Outlines Phased Deployment for OpenWay Meters in Houston." Press release (May 14, 2008). http://www.itron.com/pages/news_press_individual.asp?id=itr_016460.xml.

[20] IEEE Computer Society, LAN/MAN Standards Committee. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)." IEEE 802.15.4-2006, New York, NY, 2006. http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf.

[21] IEEE Computer Society, LAN/MAN Standards Committee. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), Amendment 1: Add Alternate PHYs." IEEE 802.15.4a-2007, New York, NY, August 2007. http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf.

[22] "Karma" http://www.theta44.org/karma/.

[23] "Kismet." http://www.kismetwireless.net/.

[24] Konings, B., Schaub, F., Kargl, F., Dietzel, S. "Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard." Proceedings of the IEEE 34th Conference on Local Computer Networks, LCN (2009)

[25] KoreK. "chopchop." http://www.aircrack-ng.org/doku.php?id=korek_chopchop.

[26] Melgares, Ricky A. "802.15.4/ZigBee Analysis and Security: tools for practical exploration of the attack surface." Dartmouth Computer Science Technical Report TR2011-689, June 2011.

[27] Northcutt, Stephen. "Network Intrusion Detection: An Analyst's Handbook." Sams, June 1999.

[28] Ptacek, Thomas and Newsham, Timothy. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection." Secure Networks, Inc., January, 1998. http://insecure.org/stf/secnet_ids/secnet_ids.html.

[29] Speers, Ryan M. "IEEE 802.15.4 Wireless Security: Self-Assessment Frameworks." Dartmouth Computer Science Technical Report TR2011-687, June 2011.

[30] Speers, Ryan and Ricky Melgares. "Api-do: Tools for ZigBee and 802.15.4 Security Auditing." http://code.google.com/p/zigbee-security/.

[31] Schiffman, Mike. "Building Open Source Network Security Tools: Components and Techniques." Wiley, October 2002.

[32] Texas Instruments. "CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver." SWRS041B, Dallas, TX, 2010.

[33] Vladimirov, Andrew, Gravrilenko, Konstantin, and Mikhailovskiy, Andrei. "Wi-Foo: The Secrets of Wireless Hacking," page 174. Pearson Education, 2004.

[34] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. "Short Paper: Reactive Jamming in Wireless Networks How Realistic is the Threat?." ACM WiSec '11. http://www.lenders.ch/publications/conferences/wisec11.pdf.

[35] Wood, Anthony D. and J.A. Stankovic. "Denial of Service in Sensor Networks." Prentice–Hall, Inc, 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.6380.

[36] Wright, Joshua. "KillerBee: Framework and tools for exploiting ZigBee and IEEE 802.15.4 networks." Version 1.0, 2010. https://code.google.com/p/killerbee/.

[37] ZigBee Alliance. "ZigBee Specification." ZigBee Document 053474r17, January 2008.