Speaking the Local Dialect: Exploiting differences between IEEE 802.15.4 Receivers with Commodity Radios for fingerprinting, targeted attacks, and WIDS evasion

Dartmouth Computer Science Technical Report TR2014-749

Ira Ray Jenkins
jenkins@cs.dartmouth.edu
Dept. of Computer Science
Dartmouth College
Hanover, New Hampshire
USA

Rebecca Shapiro
bx@cs.dartmouth.edu
Dept. of Computer Science
Dartmouth College
Hanover, New Hampshire
USA

Sergey Bratus
sergey@cs.dartmouth.edu
Dept. of Computer Science
Dartmouth College
Hanover, New Hampshire
USA

Travis Goodspeed
travis@radiantmachines.com
Straw Hat

Ryan Speers
ryan@riverloopsecurity.com
River Loop Security, LLC

David Dowd
david@riverloopsecurity.com
River Loop Security, LLC

ABSTRACT

Producing IEEE 802.15.4 PHY-frames reliably accepted by some digital radio receivers, but rejected by others—depending on the receiver chip's make and model—has strong implications for wireless security. Attackers could target specific receivers by crafting "shaped charges," attack frames that appear valid to the intended target and are ignored by all other recipients. By transmitting in the unique, slightly non-compliant "dialect" of the intended receivers, attackers would be able to create entire communication streams invisible to others, including wireless intrusion detection and prevention systems (WIDS/WIPS).

These scenarios are no longer theoretic. We present methods of producing such IEEE 802.15.4 frames with *commodity digital radio chips* widely used in building inexpensive 802.15.4-conformant devices. Typically, PHY-layer fingerprinting requires software-defined radios that cost orders of magnitude more than the chips they fingerprint; however, our methods do not require a software-defined radio and use the same inexpensive chips.

Knowledge of such differences, and the ability to fingerprint them is crucial for defenders. We investigate new methods of fingerprinting IEEE 802.15.4 devices by exploring techniques to differentiate between multiple 802.15.4-conformant radio-hardware manufacturers and firmware distributions. Further, we point out the implications of these results for WIDS, both with respect to WIDS evasion techniques and countering such evasion.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

Keywords

IEEE 802.15.4; ZigBee; wireless sensor networks; security

1. INTRODUCTION

Ubiquity of digital radios. Wireless sensor networks (WSN) represent a massive and rapidly growing technology sector. These devices will monitor and control many aspects of our daily lives, from home automation to health care monitoring to industrial management. Some market research estimates 1 billion radio frequency integrated circuit (RFIC) devices will be deployed by 2017 [29], the majority of which will be IEEE 802.15.4 [4] and ZigBee [43] standards compliant. It is estimated that by 2015, nearly 65 million digital utility meters, or "smart meters," will be installed in homes around the United States [30].

Importance of studying commodity hardware capabilities. Increasing dependence on these digital radios calls for understanding their stacks from the ground up, including the peculiarities and capabilities of their various PHY-layer commodity implementations. In particular, manipulations of PHY- and LNK-frames achievable with commodity means—and other commodity stacks' reactions to them—are of special interest for defenders. Overlooked capabilities of commodity hardware leads to nasty surprises for the defenders of a deployed base previously considered reasonably secure. For example, discovery of methods to inject arbitrary crafted 802.11 LNK-frames with commodity 802.11 hardware around 2005 led to the embarrassing "Month of Kernel Bugs" (MoKB) in 2006 that exposed multiple vulnerabilities in Wi-Fi drivers across all operating systems and in many embedded implementations. Wi-Fi suddenly became the path to Ring 0 attacks that merely required a user to open a laptop to be penetrated.[1]

In this work, we demonstrate just such effects on IEEE 802.15.4-conformant devices via PHY-layer frame shaping with commodity hardware. Our techniques impact the defensive monitoring, fingerprinting, and offensive targeting of 802.15.4 networks, including bypasses of wireless intrusion detection and prevention systems (WIDS/WIPS).

Operational importance of device fingerprinting. Active fingerprinting of wireless devices at the PHY-layer is important for operational reasons, especially when the operator suspects that rogue or "evil twin" devices may be present. The deeper the layer at which active fingerprinting operates, the tougher are the timing requirements for the attacker to imitate a particular behavioral profile logically, on non-native hardware. For mission-critical systems, such as patient insulin pumps and power grid monitors, quick-and-dirty, but accurate identification of network devices in field environments is very useful—perhaps more practically useful than the corresponding capabilities of tools like Nmap [7], Xprobe [12], or P0f [8] in enterprise networks.

Offensive implications of commodity-level PHY-receiver differences. We demonstrate that commodity 802.15.4 digital radios are capable of producing PHY-frames that differ in appearance between various 802.15.4 receivers; while being accepted as valid by some receivers, these frames may be rejected by others—depending on the radio chip's make (and, occasionally, firmware). This has direct implications for attack planning. Using this physical-frame-level technique the attacker can:

- Craft and broadcast "shaped charges," attack payloads that appear valid to only the intended target.
- Covertly communicate with target nodes by constructing a "dialect" of IEEE 802.15.4 PHY-frames that will be intelligible to only these nodes, not others of a different make or firmware. We call this covert channel technique a *deliberate partial compatibility channel*, or *dialect channel*.[2]
- Bypass WIDS/WIPS systems that utilize a different digital radio receiver, as monitor(s), than that of the network nodes they protects. This is a specific use of deliberate partial compatibility of PHY-frame dialects.

The purpose of this work is to expand the state-of-the-art in IEEE 802.15.4 physical-layer manipulation achievable with commodity 802.15.4/ZigBee devices, enabling device identification, targeted attacks, and WIDS/WIPS bypasses. We have built an experimental framework, code-named Isotope, around commodity hardware and open-source software. We have also developed several techniques that we hope to prove effective, with experimental and statistical significance, in differentiating between multiple devices' hardware and firmwares.

The remainder of this paper is organized as follows: Section 2 discusses previous work and provides context for our contributions; Section 3 introduces the offensive implications of this work; Section 4 provides a brief primer on the IEEE 802.15.4 standard and introduces the frame crafting techniques we have developed; Section 5 describes our experimental setup; Section 6 reveals our preliminary results; and, finally, Section 7 offers concluding remarks and a nod toward future work.

2. PREVIOUS WORK

Our work extends previous work on *active fingerprinting* from our lab [19, 13, 22, 40]. It also harkens back to the classic work on evading intrusion detection and prevention systems (IDS/IPS) [36, 28] that exploited differences in network streams reassembly by the attack targets and the IDS/IPS protecting them—which have since been generalized as *parser differential* attacks [32, 37].

We note that previous work in digital radio fingerprinting has focused primarily on *transmitters* rather than receivers. In constrast, we focus on fingerprinting *receivers*, which immediately delivers the attack insights that we describe in Section 3.

2.1 Digital Radio Fingerprinting

In this subsection, we briefly describe the types of digital radio fingerprinting and their application to offensive and defensive exploits. For a more detailed understanding, Danev, Zanetti, and Capkun provide a thorough survey of the state-of-the-art in wireless fingerprinting [21].

Physical-layer device identification, or fingerprinting, endeavors to exploit unique (often subtle) characteristics in the digital circuitry or firmware implementation of a device. Slight imperfections in the radio circuitry, introduced during the manufacturing process, might be detectable during radio transmissions. In addition, bugs or deviations from the standard in the firmware implementation may also be observable during radio operation. These imperfections, bugs, or deviations are known as fingerprints or device signatures.

There are both passive [23, 24, 31] and active [19, 13] methods of fingerprinting wireless radio devices. In passive methods, a third party attempts to unobtrusively sniff the communications channel. Unique signals or transmission timing may be considered a fingerprint. Naturally, this approach is often lossy or error prone due to the potential lack of traffic over the wire or interference from the multiple layers of the radio stack [36]. Alternatively, active techniques at-

[1]The title of the presentation that set off MoKB, "Hijacking a MacBook in 60 seconds," aptly captured the security community's dismay at this new attack vector; previously, the primary danger to Wi-Fi connections was believed to be sniffing of unencrypted frames.

[2]In Frank Herbert's epic science fiction novel "Dune," opposing military forces used *battle languages* unintelligible to enemies, due to shared religious proscriptions on computing systems that excluded most kinds of complex automation. We decided for a more neutral term, but recognize that it would perfectly fit a more developed version of the dialect channels we describe.

tempt to interact with a device, often by sending specially crafted requests, in hopes of eliciting a response. Both the data contained in the response and the response itself can be considered a fingerprint.

Applications of Fingerprinting. Fingerprinting digital systems has a long history of offensive and defensive applications. Security tool collections such as BackTrack Linux [1] include a growing number of fingerprinting tools, and security education organizations such as SANS [9] treat it as an essential topic.

For attackers, fingerprinting targets has long been a way of focusing effort on finding systems known to be vulnerable. It is essential in the presence of defensive misdirection measures such as false bannering or redirecting honeypots [11], as it helps to see through the defenders' deception. Not surprisingly, as soon as fingerprinting techniques became a part of standard TCP/IP network reconnaissance (in toolkits such as Nmap and Xprobe) an arms race ensued with tools such as Honeyd [3] and IP Personality [5] offering functionality to deceive fingerprinting techniques by imitating known signatures.

Impersonating trusted wireless nodes has long been a premier tool in attackers' arsenals. A tool that can identify software, firmware, or hardware and its version by highlighting differences between implementations, is especially useful when identifying wireless nodes, both benign and malignant, and finding vulnerable software, firmware, and hardware combinations. The IEEE 802.15.4 and ZigBee standards offer no exception to this rule. By design, these are commodity technologies (in particular, much more so at their origins than 802.11/Wi-Fi). Impersonating a wireless node does not pose a considerable challenge to attackers, barring strong cryptographic identification of nodes. Fully functional IEEE 802.15.4- and ZigBee-conformant digital radios can be acquired cheaply[3].

Ubiquitous deployments of IEEE 802.15.4 devices pose considerable authentication challenges [39], and it is not clear if classic PKI-based two-way authentication schemes will be a practical solution. Given the lack of strong cryptographic authentication during a device's commissioning phase [4], to be able to fingerprint an IEEE 802.15.4 radio on a device as belonging to a particular vendor's fleet may provide a piece of crucial evidence for trusting the device. Even when cryptographic authentication is in use, the implementation details of key storage and management may be problematic[5], and may lead to the keys being extracted and used by adversaries. In such situations, the capability to fingerprint physical devices may provide an additional layer of assurance when authentication material comes under suspicion.

[3]See `http://www.adafruit.com/category/29` for some example products.

[4]Some ZigBee profiles, such as the Home Automation Profile, have a defined initial key (included in the specification) which it uses to encrypt the initial key transport frame which distributes the network-wide key.

[5]For example, although the specification says that up to 255 ACL entries may be supported, some radios like the CC2420 only support 2 ACL entries [35].

It is worth noting, to the authors' knowledge, the methods we describe in this paper and their application to the IEEE 802.15.4 standard represent the state-of-the-art in wireless fingerprinting without using software-defined radios.

2.2 Parser differential attacks

The requirement for security components to interpret potentially hostile inputs the same way as the rest of the system or network they protect has long been an implicit requirement for such components' efficacy. For TCP/IP stacks, Ptacek and Newsham previously highlighted it [36], however, it was not yet made formal. The analysis of X.509 parser implementation differences [32]—leading to a plethora of attacks on the SSL Certificate Authority (CA) infrastructure using crafted Common Names interpreted differently by CAs (as low-value domain name) and SSL clients (as high-value domain name)—exposed it as a *formal* requirement for distributed systems.

Since then, such *parser differential* attacks have been generalized to many kinds of systems and protocols [37, 38], including the 802.15.4, 802.11/Wi-Fi, and similar PHY-layers [27], and recently even to 802.3/Ethernet [18].

3. ATTACK AND EVADE VIA RECEIVER FINGERPRINTING

As we noted above, digital radio fingerprinting efforts have primarily focused on fingerprinting transmitters, down to the individual radio frequency (RF) characteristics of a single radio. Such fingerprinting focus helps security models that stress authentication and attribution but do not inform defenders about "sneaky" targeted attacks that may be launched against their systems. In contrast, our active fingerprinting methods focus on weaknesses of receivers and their use by attackers.

Shaped charges for targeted receivers. Digital radios enable supervisory control and data acquisions (SCADA) applications in physical systems, such as buildings or entire neighborhoods, where the wired connectivity costs of installing sensors and remotely controlling units would be prohibitive. With 802.15.4 radios at the forefront of these SCADA applications, an attacker or a penetration tester of a SCADA wireless network may soon need to consider the presence of other radios while planning an attack or even a casual scan.[6]

For example, a "smart" building may feature several digital radio systems from different vendors, using different makes of radios, with an admixture of personal wireless devices of various degrees of criticality. Planning an attack on one of these subsystems would require avoiding accidental damage to others. Luckily, crafting the attack frames to a particular vendor's radio used by the targeted system may solve this problem. We call such attacks shaped charges, where the intended damage is limited to a particular subset of receiving systems.

[6]Penetration testing folklore has many stories of legacy systems unexpectedly crashing due to a scan, and of customers expressing such concerns during assessment planning. While hard data on such occurrences may be impossible to get, we are reminded of the popular early 90s hoax, "don't open suspicious email, it may infect your computer with a virus"; which a decade later turned into a user education mantra.

WIDS/WIPS-resistant dialects. Our work has significant implications for the design of future IDPS. At the very least, it would inform digital radio monitoring and IDPS with some clues of what to look for below the level of the logical bytes of captured frames, i.e., what attacks may be facilitated—and so also, detected and disrupted—by crafting not just the frame payloads, but also their physical layer (PHY) and physical layer convergence protocol (PLCP) representations.

Critical to defense is the knowledge of and ability to recognize WIDS/WIPS evasion. Thus, it is key to understand any stable ways in which a WIDS/WIPS may not see a frame (beyond just being out of range), but the target would. These types of attacks are introduced in the seminal work by Ptacek and Newsham [36]. They introduced injection and evasion attacks in which the data seen over the wire differed between the receiver and the IDPS.

If a frame is ignored by a receiving radio, then carefully crafted packets can be injected into the WIDS/WIPS's view of a monitored communication. On the other hand, frames that the WIDS/WIPS may ignore, but that are accepted by a receiving radio represent the potential for an entire back-channel of communications that a monitor would never see—an entire attack dialect that escapes the monitor's notice or confuses the WIDS/WIPS into judging attack communications meaningless or innocuous. We anticipate the emergence of tools that produce such dialects for digital radio protocols, just as they appeared for wired protocol features such as IP fragmentation (*fragroute*), TCP/UDP segment injection (*firewalk*), and similar tools for manipulating TCP/IP stack differences.[7]

4. METHODS

In this section, we look at the IEEE 802.15.4 standard and describe the receiver fingerprinting and targeting techniques we have developed.

4.1 IEEE 802.15.4 Standard

The Institute of Electrical and Electronics Engineers (IEEE) [4] created the 802.15 workgroup for Wireless Personal Area Networks (WPAN) in the early 2000s to establish standards for Layers 1 and 2 (physical and link, respectively). The IEEE 802.15 workgroup defined standards that include 802.15.1, a derivative of Bluetooth intended for general WPANs, and 802.15.4, designed for low-rate WPANs (LR-WPANs). LR-WPANs are attractive for low-power, low-range, low-bandwidth, and low-cost applications of wireless networking, particularly for industrial control and embedded systems.

ZigBee is a Layer 3 (network layer) specification which layers on top of the 802.15.4 layers and is more well-known than 802.15.4. While ZigBee is ripe for investigation in many different forms of fingerprinting, this paper focuses on the layer beneath ZigBee—the IEEE 802.15.4 standard.

In the IEEE 802.15.4 standard, the smallest amount of information that can be sent over the air is four bits, also known as a symbol. The standard defines four types of physical frames: beacon, data, acknowledgement, and command. The standard physical frame layout, for all four types of frames, is shown in Figure 1. A standard frame consists of a synchronization header (SHR), a physical layer (PHY) header (PHR), and a payload within the physical service data unit (PSDU). The physical frames differ in their payload, but all contain a standard SHR and PHR. The SHR comprises an 8-symbol wide preamble of zeros (0x0) and the start-of-frame delimiter (SFD), which must be 0xA7[8]. This header, as its name implies, serves to synchronize the receiving radio with the transmitting radio so that symbols are correctly pulled out of the signal. The frame length, a 7-bit number representing the number of octets in the physical payload, and a single reserved bit compose the PHR. The payload, or packet, follows the length and contains all the data for Layer 2 and higher. Each type of physical frame requires a different payload structure. Finally, not shown here, the optional 4-symbol wide frame control sequence (FCS) is a checksum used to check for data corruption in the payload during transit.

Symbols: 8	2	2		variable
Preamble	SFD	Frame length (7 bits)	Reserved	PSDU
SHR		PHR		PHY payload

Figure 1: An IEEE 802.15.4 standard physical frame. For all physical frames, the SHR should be 8 symbols of zero (0x0) followed by 0xA7. The frame length, in octets, varies with the size of the physical payload. Physical frame types differ in their payload requirements. The final element of the payload, not shown, may be the FCS.

4.2 Fingerprinting Techniques

Here we will describe four new techniques for fingerprinting IEEE 802.15.4 stacks, with a focus on the physical layer. Each technique is active—a stimulus frame with a non-standard physical-layer header is transmitted and the target's response or lack thereof is recorded. Our hypothesis is that we can distinguish different radio chipsets by which type of stimulus packets they are able to receive. To determine whether a given chipset has indeed received a packet, we send a frame whose payload triggers a response by a higher layer—such as beacon request. If we receive the correct response to our stimulus, we assume that our crafted frame was received.

Crafting Physical Frame Headers. Before introducing the designed methods, it should be noted that many commodity radios cannot craft arbitrary physical frame headers, SHR and PHR. By design, the radio hardware manages the frame headers to assure proper functionality. In order to fully control a physical frame's contents, we make use of our good neighbor Travis Goodspeed, et al's packets-in-packets (PIP) frame-injection technique [27].

The PIP technique for IEEE 802.15.4 digital radios is relatively simple. The 802.15.4 standard requires the SFD to be 0xA7. If an 802.15.4-compliant radio receives an SFD of any

[7]For examples see, "Building Open Source Network Security Tools: Components and Techniques", Mike Schiffman, Wiley 2002.

[8]Some radios deviate from the standard and allow the SFD to be set via an internal register.

other value, the receiving radio resets itself into a fresh receiving state, listening again for a new SHR. As noted above, some radios permit us to specify the SFD value via a register, which allows us to transmit frames with non-compliant SFDs. Any receivers expecting the standard SFD will reset themselves after seeing the unexpected symbols. The transmitting radio, however, will continue to send the remainder of the frame. If the remainder of the frame contains a standard SHR the receiver will think it is receiving a fresh packet. In this way, we are able to transmit a non-standard physical frame that contains a fully-standard physical frame, a packet in a packet.

A Variable Preamble Length. The variable preamble length fingerprinting technique focuses on the preamble used to put the receiving radio into a state where it is ready to accept an SFD followed by the remainder of the frame. While the IEEE 802.15.4 standard defines the preamble length to be eight symbols containing the value 0x0, some radios might accept frames with fewer than the stated number, while others do not. Figure 2 shows the general layout of a frame generated to test a target's response to non-standard preambles. The aim of this technique is to measure the number of zero symbols, before the SFD, a chipset requires in order to accept a frame. Note that the only portion of the frame that is altered from the IEEE 802.15.4 standard is the preamble length.

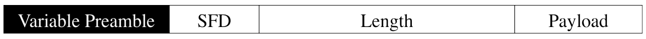

Figure 2: A physical frame with variable preamble length. The number of zero (0x0) symbols that compose the preamble is varied between 0 and 8. Any response to a non-standard preamble might signify a fingerprint.

A Franconian Notch. According to the IEEE 802.15.4 specification, a preamble field should contain 32 binary zeros—eight zero (0x0) symbols. However, some chipsets may accept non-standard preambles. For example, the CC2420 [41] can be programmed to ignore some of the least significant symbols in the synchronization header to help it be more resilient to noise. Figure 3 shows the physical frame crafted for the Franconian Notch[9] method. Here we modulate each subsequent symbol of the standard preamble from 0x0 to 0xF[10], going from all zeros (0x0s) to all 0xFs. The aim of this technique is to measure the number of invalid preamble symbols a radio is willing to accept. Note, again, that the only portion of the frame that is modified from the IEEE 802.15.4 standard is the preamble symbols.

A Franconian Bridge. Inspired by the previous approaches, the Franconian Bridge method "spans the gap" between the variable preamble length and Franconian Notch techniques. As shown in Figure 4, the Franconian Bridge checks to see how a target responds to having a varying number of 0xF symbols placed between the fully-standard preamble and the SFD. Technically, this will evaluate a radios behavior in the

[9]The Franconian Notch is a mountain pass through the White Mountains of New Hampshire.

[10]It should be noted that we do not attempt to modulate all of the possible combinations of 0x0s and 0x1s.

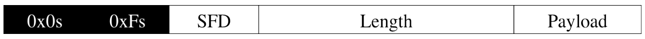

Figure 3: A physical frame with Franconian Notch. The number of zero (0x0) symbols that compose the preamble is varied between 0 and 8, with the lengths remainder transformed into 0xF symbols. Any response to a non-standard preamble might signify a fingerprint.

presence of a seemingly non-standard SFD. As before, the only portion of the frame that is modified from the IEEE 802.15.4 standard is that which follows the preamble and precedes an SFD.

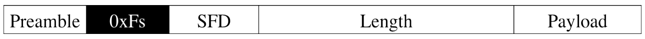

Figure 4: A physical frame with Franconian Bridge. A varying number of 0xF symbols are inserted between a fully-standard preamble and SFD. Any response to a non-standard SFD might signify a fingerprint.

A Cumberland Gap. The Cumberland Gap[11] technique, as seen in Figure 5, measures how a target behaves with respect to receiving frames immediately after receiving a valid preamble and an invalid SFD, followed by a standard frame.

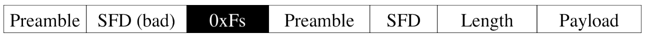

Figure 5: A physical frame with Cumberland Gap. An invalid SFD is injected, followed by a varying amount of garbage symbols. Any unique response might signify a fingerprint.

It is important to remember that when radios are listening for data, they read whatever they find into a symbol. Therefore, it is quite common for a radio to be prepared to accept a frame when it is merely listening to interference and reading garbage as symbols. There are a few discrete states that a radio state machine has to go through when finding an SFD. In this method, we intentionally make the SFD very close to the standard to nudge the receiver as close as possible to the state in which it receives a full frame without outright telling it to take the remainder of the frame. When the incorrect SFD arrives, the chip goes back to listening for a preamble—we seek to measure the timing of this behavior. The fewer symbols that we can inject and still get a response may imply a faster turn-over time, and might also signify a fingerprint.

5. EXPERIMENTAL SETUP

To test the functionality of our proposed fingerprinting methods, we built a testbed to examine how different IEEE 802.15.4 stacks respond to the types of non-standard physical headers previously described.

5.1 Testbed Layout

[11]The Cumberland Gap is a mountain pass through the Appalachian Mountains between Tennessee, Kentucky, and Virginia.

Our testbed consists of only commodity hardware and open-source software. As shown in Figure 6, two IEEE 802.15.4-conformant radios are connected (via serial over USB) to a single workstation running Isotope, our fingerprinting software. Isotope is a Python framework that utilizes the open source libraries Scapy [10], to build 802.15.4 physical frames, and KillerBee [6], to configure the radios, monitor communications traffic, and inject arbitrary frames. One radio is used solely to transmit crafted frames and the other radio is used to sniff all traffic on a particular channel. The third, unknown, device is setup to listen on a specific channel and respond to beacon requests.

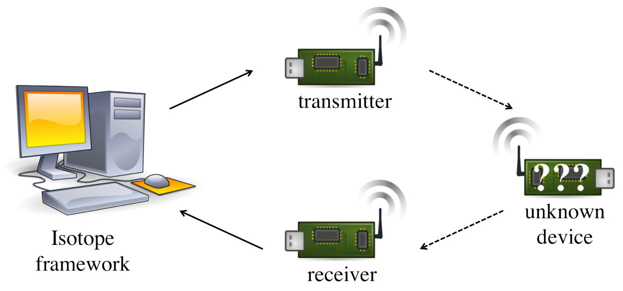

Figure 6: The fingerprinting testbed. Our Python framework, Isotope, manages separate transmitting and receiving radios and monitors communications. All radios operate on the same channel, with the transmitter sending out non-standard beacon requests. The unknown device is configured to listen for valid requests and respond. The receiving radio listens for beacon responses.

Although this setup may appear contrived—802.15.4 devices may be configured to hop between various channels, as they send and receive frames, for additional robustness or to frustrate reverse engineering—we believe that our setup is a good starting point and that it can be extended to work with a variety of target configurations.

5.2 Hardware and Software

We tested multiple devices[12] including Zigduinos [33], RZUSBsticks [17], and the popular (but now discontinued)Tmote Sky [34]. Each of these devices contain different on-board radio chips, namely an Atmel ATmega128RFA1 [16], an Atmel AT86RF230 [15], and a Chipcon CC2420 [41], respectively. Finally, each device has several associated open-source firmware distributions including Arduino [14], Chibi [25, 26], Contiki OS [20], GoodFET [2], and Tiny OS [42]. Table 1 summarizes the different possible combinations.

6. RESULTS

To-date, only the GoodFET firmware combinations have yielded results. The following charts represent the individual beacon responses received, out of 1000 non-standard beacon requests, for each radio device with GoodFET firmware.

A variable preamble length. Figure 7 shows the results of varying the number of preamble symbols from 0 to 7—8 zero (0x0) symbols is the standard. Clearly, the Tmote

[12]We did not test the Freakduino, and only recently received the Api-Mote to replace the Tmote Sky.

Device	Radio	Firmware
zigduino	atmega128rfa1	Arduino Contiki GoodFET TinyOS
freakduino	at86rf230	Arduino Chibi
rzusbstick	at86rf230	Chibi Contiki Raven
apimote	cc2420	GoodFET
tmote sky	cc2420	Contiki GoodFET TinyOS

Table 1: Hardware and Firmware Combinations

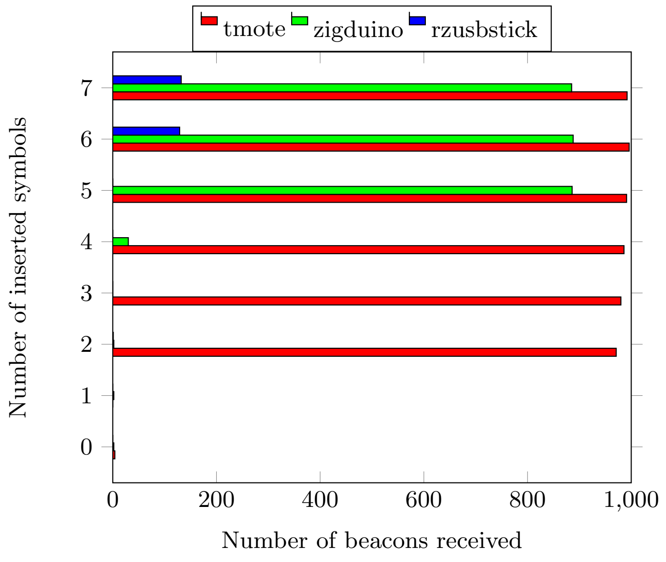

Figure 7: Variable preamble results.

device responds to the fewest number of preamble symbols. It is possible that this is by design. Remember, the Tmote contains a CC2420 radio chip which allows a programmable number of preamble bits to be accepted. Assuming normal function, it seems obvious that the Tmote is distinguishable from the Zigduino and RZUSBstick. Somewhat unsettling is that the RZUSBstick responds to less than 200 beacon requests with 6 or 7 symbols. It is possible that this device is more strictly-standards compliant; a test run with exactly 8 symbols would verify this assumption.

A Franconian Notch. Figure 8 showcases the results of transforming the preamble from 8 zero (0x0) symbols to 6 0xF symbols. Zero (0) on the Y-axis represents a fully standard physical frame, with zero 0xF symbols present. It appears as though the Tmote, previously loose with the standard, is now fully compliant. Since the Tmote previously accepted fewer preamble symbols, this could be an artifact of the radio interpreting the additional 0xF symbols as an invalid SFD, or it could have to do with the RF demodulator's sync circuit being thrown out of state by the additional bit transitions. Again, the RZUSBstick responds to far fewer

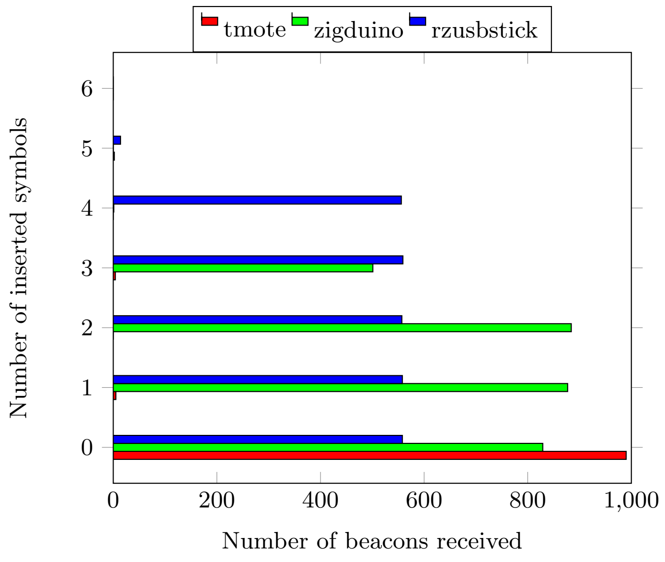

Figure 8: Franconian Notch results.

beacon requests. This may be explained by the position of the mote during testing or the fact that both the Tmote and Zigduinos use external antennas. In either case, the RZUSBstick stands out by accepting as many as 4 0xF symbols within the preamble. For both the Tmote and RZUSBstick, this looks like a possible identifier.

To verify the integrity of these promising results, we re-ran the Franconian Notch on even symbols with an RZUSBstick and, in place of a Tmote, a newly acquired ApiMote. The results from this limited test are shown in Figure 9. The RZUSBstick was determined as having received a frame if it produced the requisite beacon frame in response (results also confirmed with it's logs and the ApiMote's PCAP), and the ApiMote collected full PCAP which was processed to count how many times it captured the Beacon Request frame depending on which Franconian Notch preamble variation was used.

Of particular note is that the RZUSBstick preforms remarkably well receiving packets with up to 6 symbols of 0xF before the SFD. However, the ApiMote's CC2420 chip [13] did not receive packets proceeded with a preamble which did not have 8 symbols of 0x0. Both of these results seem consisten with our earlier tests, and even tacitly suggest that the ApiMote is a good successor to the Tmote.

A Franconian Bridge. The results for the Franconian Bridge method are shown in Figure 10. Recall that this technique inserts garbage between a valid preamble and a valid SFD. Ideally, a radio would interpret the garbage as an invalid SFD. As in the previous method's results, the Tmote strictly adheres to the standard; while, the RZUSBstick drastically increases its responses from the previous two tests. The RZUSBstick accepts up to 5 garbage symbols interposed between the preamble and SFD.

[13]The ApiMote was running the GoodFET firmware, with the standard values unchanged in the CC2420 for both `MDMCTRL0.PREAMBLE_LENGTH` and `SYNCWORD`.

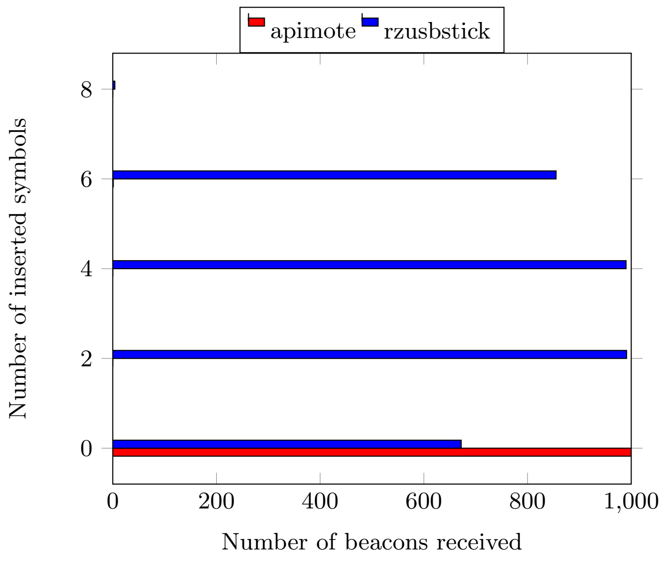

Figure 9: Franconian Notch results on even-sized symbols.

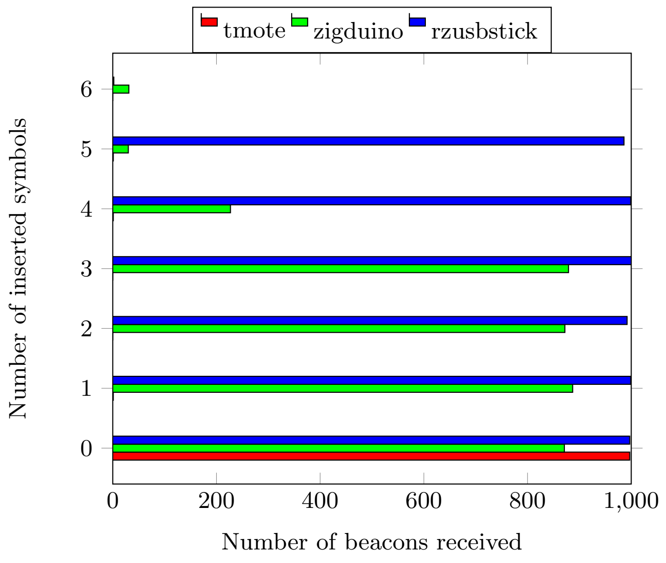

Figure 10: Franconian Bridge results.

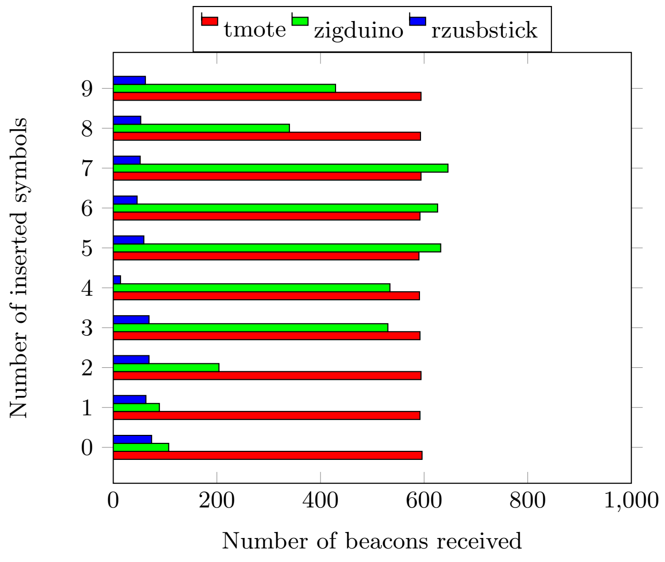

Figure 11: Cumberland Gap results.

A Cumberland Gap. The results for the Cumberland Gap method, seen in Figure 11, do not seem encouraging. None of the motes respond to more than about 600 beacon requests. There may have been some interference or channel noise during this test run. Additional tests should be performed. It appears as though the Tmote has the fastest turnaround time, while the RZUSBstick maintains the slowest.

7. CONCLUSIONS

With the number of wireless sensor networks exploding, a large portion being IEEE 802.15.4 and ZigBee devices, it is essential that we be able to secure and protect these devices and networks for mission-critical systems. Fingerprinting these radio devices is a first step along the path to achieving that security. Device identification, both passive and active, has been used on many other wireless network protocols. Our work seeks to apply it to IEEE 802.15.4-conformant radio devices. By accurately identifying different devices, we have another tool, on-top of PKI authentication schemes, for verifying trusted nodes in a network. Similarly, by analyzing how frames and packets make their way through the firmware and radio circuitry, it is possible that we may uncover hidden vulnerabilities and attack vectors.

With only preliminary results, it appears that the Tmote devices (and new ApiMote devices), with the Chipcon CC2420 radio chipset, and the RZUSBsticks, with the Atmel AT86RF230 radio chipset, are identifiable. A summary of our results to-date is shown in Table 2. The Tmotes clearly respond to very non-standard preamble lengths, whether by design or flaw; however, the same devices seem to be very strict on preamble and SFD content. Meanwhile, the RZUSBsticks present a conundrum. In three of the tests, the devices respond with an alarmingly low rate. It is possible the devices are very slow, are receiving too much noise, or simply do not receive all the beacon requests without external antennas. From the results that we do have, it looks like the RZUSBsticks accept very non-standard preamble and SFD content. The CC2420 chips look like the top contender so far to avoid WIDS detection. Further work would need to confirm this.

	Firmware	Preamble	Franconia Notch	Franconia Bridge	Cumberland Gap
ATmega128RFA1	Contiki	—	—	—	—
	Goodfet	H	H	H	H
	TinyOS	—	—	—	—
	Zigduino	H	H	H	H
AT86RF230	Chibi	—	—	—	—
	Contiki	—	—	—	—
	Raven	I	I	I	I
CC2420	Contiki	—	—	—	—
	Goodfet	H	L	L	L
	TinyOS	H	L	L	L

Table 2: Summary of results to-date. The following table is labelled based on our current confidence in identifiability. An 'H' means that for a specific radio/firmware combination, the given fingerprinting technique is likely to be distinguishable from other radio/firmware combinations. An 'L' means that, at this time, little evidence suggests a specific radio/firmware combination is easily identifiable. An 'I' means inconclusive at this time. An '—' means that we have yet to test this radio/firmware combination.

7.1 Future Work

We feel this work is ripe for research. As shown above, there are many more possible firmware and hardware combinations to test—we have really only just begun. Moving forward, it will be necessary to evaluate the effect of noise and interference on the testbed. Of course, our software framework, Isotope, will also require some additional refinements to make it more robust. Typically, in device identification, a database of fingerprints is used in combination with some sort of machine learning method to analyze and evaluate fingerprint matches. Our current work constitutes only the first stage of identifying possible fingerprints. We should also consider additional techniques for fingerprinting, such as length overflow. Lastly, we would like to explore the potential for WIDS evasion by these commodity radios.

8. ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation, under Grant Award Number 1016782, and the Department of Energy, under Grant Award Number DE-OE0000097. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

9. REFERENCES

[1] BackTrack Linux. `http://www.backtrack-linux.org`.

[2] GoodFET. http://goodfet.sourceforge.net.

[3] Honeyd. http://www.honeyd.org.

[4] IEEE Computer Society. http://www.ieee.org.

[5] IP Personality. http://ippersonality.sourceforge.net.

[6] KillerBee. http://code.google.com/p/killerbee/.

[7] Nmap Security Scanner. http://nmap.org.

[8] p0f. http://lcamtuf.coredump.cx/p0f3/.

[9] SANS. http://www.sans.org.

[10] Scapy. http://www.secdev.org/projects/scapy/.

[11] Tiny Honeypot. http://freecode.com/projects/thp.

[12] Xprobe. http://xprobe.sourceforge.net.

[13] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the 3rd ACM conference on Wireless network security*, pages 169–174. ACM, 2010.

[14] Arduino. Arduino development platform. http://www.arduino.cc.

[15] Atmel Corporation. AT86RF230 datasheet. http://www.atmel.com/Images/doc5131.pdf.

[16] Atmel Corporation. ATmega128RFA1 datasheet. http://www.atmel.com/Images/doc8266.pdf.

[17] Atmel Corporation. RZUSBstick. http://www.atmel.com/tools/RZUSBSTICK.aspx.

[18] A. Barisani and D. Bianco. Fully Arbitrary 802.3 Packet Injection: Maximizing the Ethernet Attack Surface. BlackHat USA, July 2013.

[19] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the first ACM conference on Wireless network security*, pages 56–61. ACM, 2008.

[20] Contiki Project. Contiki OS. http://www.contiki-os.org.

[21] B. Danev, D. Zanetti, and S. Capkun. On physical-layer identification of wireless devices. *ACM Computing Surveys (CSUR)*, 45(1):6, 2012.

[22] D. D. Dowd. *Isotope: Active Behavioral Fingerprinting of IEEE 802.15.4 Devices*. Senior honors thesis, Dartmouth College, Computer Science, August 2012.

[23] J. P. Ellch. Fingerprinting 802.11 devices. Master's thesis, Naval Postgraduate School, 2006.

[24] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of the 15th USENIX Security Symposium*, pages 167–178, 2006.

[25] Freaklabs. Chibi Arduino port. http://www.freaklabs.org/index.php/chibiArduino.html.

[26] Freaklabs. Chibi wireless stack. http://www.freaklabs.org/index.php/Chibi-A-Simple-Open-Source-Wireless-Stack.html.

[27] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers. Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios. In *5th USENIX Workshop on Offensive Technologies (WOOT)*, pages 54–61, August 2011.

[28] M. Handley, V. Paxson, and C. Kreibich. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the USENIX Security Conference*, 2001.

[29] M. Hatler, D. Gurganious, and C. Chi. 802.15.4 & ZigBee: Expanding markets, growing threats. Technical report, A Market Dynamics report (9th edition), 2012.

[30] Institute for Electric Efficiency. Utility-scale smart meter deployments, plans, & proposals. Technical report, The Edison Foundation, May 2012.

[31] S. Jana and S. K. Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. *Mobile Computing, IEEE Transactions on*, 9(3):449–462, 2010.

[32] D. Kaminsky, L. Sassaman, and M. Patterson. PKI Layer Cake: New Collision Attacks Against The Global X.509 CA Infrastructure. Black Hat USA, August 2009.

[33] Logos Electromechanical LLC. Zigduino. http://logos-electro.com/zigduino/.

[34] Moteiv Corporation. Tmote Sky datasheet. http://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote_sky_datasheet.pdf.

[35] D. W. Naveen Sastry. Security considerations for ieee 802.15.4 networks. In *In WiSe âĂŹ04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 32–42. ACM Press, 2004.

[36] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks Inc, 1998.

[37] L. Sassaman, M. L. Patterson, S. Bratus, and M. E. Locasto. Security Applications of Formal Language Theory. *IEEE Systems Journal*, 7(3), September 2013.

[38] L. Sassaman, M. L. Patterson, S. Bratus, and A. Shubina. The Halting Problems of Network Stack Insecurity. *USENIX ;login:*, 36(6), December 2011.

[39] S. W. Smith. Cryptographic scalability challenges in the smart grid. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–3, 2012.

[40] R. Speers. IEEE 802.15.4 Wireless Security: Self-Assessment Frameworks. Technical Report TR2011-687, Dartmouth College, Computer Science, Hanover, NH, June 2011.

[41] Texas Instruments. CC2420 datasheet. http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[42] TinyOS Alliance. Tinyos. http://www.tinyos.net.

[43] Zigbee Alliance. ZigBee. http://www.zigbee.org.