Packets in Packets:
Orson Welles' In-Band Signaling Attacks for Modern Radios

Travis Goodspeed
University of Pennsylvania

Sergey Bratus
Dartmouth College

Ricky Melgares
Dartmouth College

Rebecca Shapiro
Dartmouth College

Ryan Speers
Dartmouth College

Abstract

Here we present methods for injecting raw frames at Layer 1 from within upper-layer protocols by abuse of in-band signaling mechanisms common to most digital radio protocols. This packet piggy-backing technique allows attackers to hide malicious packets inside packets that *are permitted* on the network. When these carefully crafted Packets-in-Packets (PIPs) traverse a wireless network, a bit error in the outer frame will cause the inner frame to be interpreted instead. This allows an attacker to evade firewalls, intrusion detection/prevention systems, user-land networking restrictions, and other such defenses. As packets are constructed using interior fields of higher networking layers, the attacker only needs the authority to send cleartext data over the air, even if it is wrapped within several networking layers.

This paper includes *tested examples* of raw frame injection for IEEE 802.15.4 and 2-FSK radios. Additionally, implementation complications are described for 802.11 and a variety of other modern radios. Finally, we present suggestions for how this technique might be extended from wireless radio protocols to Ethernet and other wired links.

1 Introduction

This paper presents methods for remote frame injection by abusing in-band signalling mechanisms that are common to most digital radios at the physical layer, Layer 1. In stark contrast to prior injection techniques, we consider the case in which an attacker has *no physical access* to the radio network and *no root access* to the machines on that network. Rather, we suppose that she only has the ability to send data within higher-layer packets that are sent through the radio network. As will be demonstrated in later sections, such injections are possible by leveraging radio symbol errors and the attacker's ability to construct her data payload as if it were a complete and valid frame in its own right. When the beginning of the outer frame is damaged due to interference, signal strength, or tuning problems, the inner frame will be interpreted as a packet, rather than a payload.

Why concentrate on injection? Packet injection has always been a major theme in both attack development and vulnerability assessments. The capability to inject messages into the medium used by a network enables many kinds of attacks on the network's nodes. In fact, many attack toolkits are built around libraries that provide and streamline injection such as *libnet* and *LORCON* [2]. Breakthroughs in the development of these toolkits have been associated with tools such as *airjack* or *KillerBee*. These tools brought reliable injection functionality to affordable commodity hardware.

The underlying reason as to why packet injection has always been a fruitful attack method is that many network stack and protocol implementers make de-facto trust assumptions regarding the origin and integrity of the headers and data. For example, the protocol implementers may assume that some protocol fields in the lower network layers cannot have their values forged or crafted without costly and rare special-purpose equipment[1] or that packets with certain values in their fields would never reach the network due to upstream or border filtering. We show how both of these assumptions, especially the latter assumption, can backfire even when private wireless networks are protected by upstream filters or by a physical Faraday cage inaccessible to the attacker.

Injection across OSI layers is bad news for defenders. Our injection method acts across the conceptual

[1] Arguably, such assumptions on the part of the 802.11 kernel driver developers were at least partly to blame for the prevalence of 802.11 drivers in the so-called "Month of Kernel Bugs" [9] ushered by the "ring 0" lax link-layer data handling exploit [8].

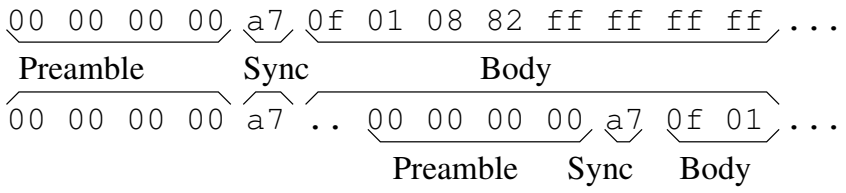

Figure 1: Three-part 802.15.4 frames with (bottom) and without (top) a PIP.

boundaries of the OSI network model's layers. Injection of crafted Layer 2 (the Data Link layer) frames is achieved by transmitting a well-formed packet with a crafted Layer 3 (the Network layer) or higher payload, as shown in Figure 2. The attacker need not alter the packet's enclosing Layer 2 structure provided by the standard network stack; the stack is simply relied upon to normally wrap and transmit the crafted Layer 3 or higher payload. The transmitting node's Layer 2 and Layer 1 (the Physical RF layer) network stack logic will "believe"[2] that it is transmitting a normal Layer 3 or higher payload. However the crafted packet will trick nodes into receiving the crafted Layer 2 frame.

Historically, wireless Layer 2 injection techniques have been associated with the attacker manipulating (that is, compromising integrity of) code or configuration data responsible for building Layer 2 headers. Since the logic that builds Layer 2 headers is typically a part of the operating system (e.g., resides in a kernel driver) and the integrity of the configuration data is protected by the kernel, the attacker is commonly assumed to need *at least* superuser-type control of the node, a kernel compromise, or a similar achievement to succeed at Layer 2 injection. Simply put, only those attackers who could "mess with kernel or drivers" were expected to succeed at Layer 2 injection. Our work demonstrates this presumed connection to be a fallacy: attackers can perform Layer 2 injections by shaping Layer 3 and above traffic. Such injection can be done *without* a kernel compromise or superuser privileges in an internal node. In fact, through the use of forwarded application data, the technique can be leveraged without *any* internal node being compromised (as we discuss in Section 3).

Does node hardening protect against injection? The assumption that certain kinds of injection require a *stronger-than-userland attacker achievement*, such as a superuser account compromise or "ring 0" access, may lead to policies where the goal of protecting the integrity of internal WLANs is meant to be fulfilled by strong system hardening measures. These hardening measures might be expected to contain (inevitable) application compromises to prevent the attacker from abusing Link layer protocols that do not support message authentication and are trusted at face value.

This mistake is particularly easy to make in the presence of Mandatory Access Control, virtualization-based isolation, security hypervisors, application sandboxing, or Trusted Computing-based mechanisms (e.g., Trusted Network Connect with mandatory attestation of the network nodes) that constrain the power of the superuser, protect the integrity of the OS, and are intended as a barrier to local privilege elevation attacks.

In this paper we demonstrate that our Packet-in-Packet injection technique invalidates such assumptions on affected radio networks. User-level application compromise and the resulting ability to generate well-formed Layer 3 and higher traffic in a wireless network can result in an arbitrary Layer 2 crafted packet injection capability, in stark contrast with Ethernet and other wired data link layers.

2 Packets in Packets

Our technique for raw frame injection works by placing a complete radio frame within the body of a larger frame, then leveraging noise or protocol differences to cause the start of the outer frame to be missed. Once this happens, the receiving radio will continue to process the start of the outer packet as if it were background noise. Upon reaching the interior packet, the receiver—thinking this to be the start of a unique packet—will interpret it as a packet in its own right, rather than as data within an upper-layer protocol. In this way, it is possible to remotely inject raw packets without physical proximity or root access.

As shown in the upper half of Figure 1, digital radio physical layer protocols generally consist of three distinct regions: (1) a Preamble, (2) a Sync, and (3) a Body. The **Preamble** serves to wake up the radio, indicating that a message might follow. The **Sync** field is usually between two and five bytes long. It serves both to synchronize the clock of the transmitter with that of the receiver and to distinguish legitimate packets from background noise. The **Body** carries the rest of the packet,

[2]That is, there is no tampering, and short of special scanning for our payloads, no internally visible evidence of the stack being used in an injection attack.

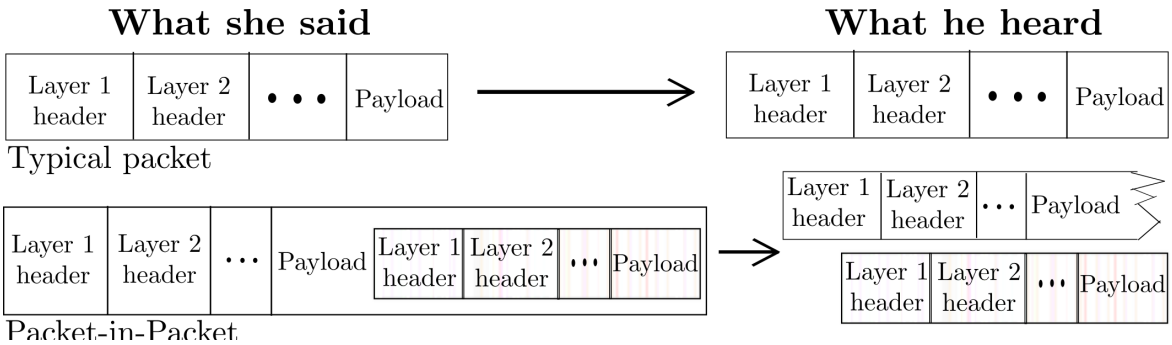

Figure 2: A typical packet's interpretation contrasted with that of a PIP.

including data from the upper layers. Radios encode the entire frame into a series of **symbols** for the purpose of transmission. With a few exceptions,[3] the three parts use the same symbol set, allowing the Preamble and Sync values to be constructed within the Body. There are multiple ways to encode data onto a carrier wave including frequency shift keying (FSK) and phase shift keying (PSK). **FSK** encodes data by modulating the frequency of the carrier wave, **PSK** encodes data by modulating the phase.

Our Packet-in-Packet (PIP) technique consists of crafting a complete three-part frame within the body of a higher-level packet, which is itself contained within a legal outer frame. That is to say, a radio frame that would be valid on its own is placed within the data portion of a larger frame. This is shown in the lower half of Figure 1. When there is a symbol error that causes the first, legitimate Sync to be missed, a receiver will see the inner packet's payload as if it were a complete and raw radio frame.[4] In this way, an attacker who controls any field within a high-layer packet—such as a DNS query response or an HTTP page—can inject a raw radio frame into a remote wireless network.

Unlike interpretation of a digital file, symbol errors within digital radio packets *can and do occur* with surprising frequency. An attacker who wishes to inject a packet can rebroadcast his attempt until the inner packet—rather than the outer—is seen. Section 5 describes a number of complications which must be taken into account when implementing this sort of injection, while Section 4 presents concrete examples for IEEE 802.15.4 and 2-FSK radios.

[3] As described in Section 5, some variants of 802.11 change symbol sets in the middle of a packet in order to support fast data rates while maintaining backward compatibility. As the body is encoded with a superset of the header symbols, a valid header can often still be produced within the body.

[4] Sometimes the Sync is missed for other reasons, as in the case of injection between protocols or injection in those protocols which use the Sync as a destination address.

3 Threat Scenarios

Attacks utilizing this technique can come in a variety of flavors, which are enumerated here by means of injection. This list is by no means complete, and it is expected that many new styles of attacks will stem from this line of research. In each case, we show how a malicious actor, Mallory, can perform otherwise prohibited acts by way of packet-in-packet injections.

The most obvious use of PIP injection would be to remotely inject packets that would otherwise be filtered by a firewall or router. For example, Mallory might have Alice download a large file composed of 802.11b beacon frames. If Alice downloads the file from a coffee house's unencrypted 802.11b network, patrons in that venue will see a false access point with an SSID of "Mallory was here." or "Call +1.555.555.5555."

In the case of broadcast packets such as beacon announcements, Mallory can prepare a single file before the attack, rather than custom tailoring it to Alice's network. For more specific injections, Mallory might have an unprivileged account on the machine communication is established with and wish to inject traffic using that knowledge of the LAN. Mallory could use standard Unix commands to determine the MAC and IP addresses of hosts on a remote wireless network, then use PIP injection to forge local UDP packets that her lack of system privileges would otherwise prohibit. In some cases, this might even be possible on more restricted environments such as those found in Android and iOS.

Cross device injection is also a potential threat. Mallory can leverage her control of one radio device to inject traffic into an entirely different device. Because the ANT+ [1] protocol and a popular brand of classroom survey radios use the symbol set with different Sync values [3], Mallory could leverage the privilege of sending ANT+ traffic to inject fake survey responses. Because the Sync values differ, survey radio receivers will reliably ignore the outer frame and packet error rates on the PIP can approach zero. An example of this is presented

in Section 4.2.

4 Concrete Examples

In this section, we provide implementation details and tested packets for both the IEEE 802.15.4 protocol and a common 2-FSK radio. These demonstrate both the probabilistic nature of the attack when used with devices sharing a common Sync, such as 802.15.4, and the reliable outcome of the attack when applied to protocols with a varied sync, such as ANT+.

4.1 802.15.4, ZigBee

Outer	Hex	Inner
Preamble	`00 00 00 00`	
Sync	`a7`	
Body	`19`	
	`01 08 82`	
	`ca fe ba be`	
	`00 00 00 00`	Preamble
	`a7`	Sync
	0a 01 08 82 ff ff ff ff c9 d1	Body
	`15 e8`	

Figure 3: 802.15.4 PIP

IEEE 802.15.4 is a perfect platform for prototyping this technique, as it is reasonably standardized[5] across competing protocols and equipment with low-level register access is easily obtained. As there are four bits per symbol, payload data needs to be nibble-aligned. Furthermore, the standard Sync[6] value of `a7` and the de jure requirement of the 802.15.4 standard that the packet be ignored for the length of its body, it is necessary that the Sync or Body Length of the outer packet be misinterpreted by the receiver [6, 44]. Since there is no error correction for these fields, such symbol misinterpretations happen with sufficient frequency for a successful attack.

For example, `00 00 00 00 a7 0a 01 08 82 ff ff ff ff c9 d1` is a short 802.15.4 packet with a valid checksum to the broadcast PAN and MAC. In this case, `00 00 00 00` is the Preamble, `a7` is the Sync, and `0a 01 08 82 ff ff ff ff c9 d1` is the Body, consisting of fields for Length, Header, PAN, MAC, and Checksum.

Consider the case of a much longer packet shown in Figure 3 being received by a radio with a PAN of `de ad` and a MAC of `be ef`. This packet, being addressed to PAN (personal area network) `ca fe` and MAC `ba be`, should be ignored by `be ef` as the addresses do not match. Further, `be ef` should also wait until the duration of the packet has passed before returning to the listening state in which a Sync might begin a new packet. That is to say, during proper reception, the following packet's Body will not be misinterpreted for being a complete frame so long as every symbol is correctly observed by the receiver.

In order to inject the inner packet as a raw frame, the attacker must cause a packet like the one above to be transmitted multiple times, then bank upon interference damaging the Sync field of the outer packet. Supposing that an A-symbol is swapped for an F-symbol in the outer Sync, it will be seen as `f7` and the first valid Sync field will be seen as the `a7` within the body of the outer packet.

4.2 nRF24L01+ and 2-FSK

Outer	Hex	Inner
Preamble	`55`	
Sync	`01 02 03 02 01`	
Body	`55`	Preamble
	`12 34 56`	Sync
	ff ff ff 35 CK[1] CK	Body
	`CK CK`	

Figure 4: nRF24L01+ PIP

[1] The symbol "CK" represents a byte of the checksum that would be correctly calculated over the relevant section of the packet, as specified by the protocol being used.

The nRF24L01+ is a 2-FSK radio chip from Nordic Semiconductor used in both the ANT+ standard and vendor-proprietary protocols, such as those used by Microsoft, Logitech, and Hewlett-Packard wireless keyboards and mice. [10] Similar chips in the same family and competing chips from other vendors use compatible encoding schemes that vary only by data rate. As both the ANT+ protocol and the Microsoft 2.4GHz keyboards use the Sync field as a destination address, exploitation of a receiver other than the one that a packet is addressed to can be performed deterministically, with no dependence upon luck or radio noise.

Consider the PIP in Figure 4.2, in which the Sync field doubles as a destination address and both protocols are of fixed length. We will also assume, for the sake of simplicity, that both protocols are running at the same rate, although the rate corrections described in Section 5 can be applied whenever the transmitter's rate is higher than that of the inner-packet receiver. In this case, the attacker is broadcasting a packet through Protocol Foo, in which the Sync is defined to be `01 02 03 02 01`. The attacker's payload is intended for Protocol Bar on the same

[5] Standardization, in this context, should not be confused with interoperability.

[6] IEEE 802.15.4 refers to the Sync as the start-of-frame delimiter (SFD), but we shall refer to it as the Sync for consistency.

channel, but with a Sync of `12 34 56`. Because these Syncs differ, devices configured for Protocol Bar *cannot know* that a transmission intended for another recipient is already in progress. It will reliably observe and interpret the inner packet, without the rebroadcasts that are necessary in other PIP examples.

5 Complications

Just as SQL injection is made more difficult by naive character escaping, PIP frame injection is complicated by properties unique to each encoding scheme. This section describes some of these complications, as well as the techniques with which they can be surmounted.

Sync and Address While a few protocols use the Sync field as a destination address, the most common complication of the PIP technique is that the Sync field is shared by all devices within the protocol. In this case, the inner packet will be ignored during most receptions, but it will be interpreted if either (1) the outer Sync field is misinterpreted or (2) a variable-length field within the Body is misinterpreted to be smaller than its intended value. In the first case, a receiver will not know that a packet has already begun, while in the second case, a receiver will think that the prior transmission has ended. The likelihood of these events increases in inverse proportion to the link quality, and repeated rebroadcasts can be leveraged to guarantee success for the attacker.

Rate Variance Some devices, particularly chips implementing variants of frequency shift keying (FSK), offer multiple data rates with otherwise identical encoding. To inject from a transmitter at n baud into a receiver of $n/2$ baud, simply correct for the slower interpretation of the symbols by duplicating each symbol. That is to say, `80 01` at 1 Mbps in 2-FSK is identical to `c0 00 00 03` at 2 Mbps.

Encoding Variance A complication unique to protocols with large packets supporting multiple data-rates is support for mid-packet shifts in the encoding scheme. For example, IEEE 802.11b always encodes the Sync at 1 Mbps DBPSK (Differential Binary Phase-Shift Keying), then optionally changes to DQPSK (Differential Quadrature Phase-Shift Keying) at a higher data rate for the start of the header and again at the start of the payload. An attacker injecting into Wifi must therefore know (or correctly guess) the rate in use during the part of the Body that she controls. More drastic changes in encoding might be uncorrectable, necessitating trickier workarounds.

Time and Code Division Time and code division protocols, such as those used in GSM and CDMA telephones, are used to coordinate access to a shared transmission medium between multiple nodes. Such media access control (MAC) protocols can be quite difficult to work around. In particular, there is little utility in broadcasting a malicious packet when the intended recipient is not listening or is using a code scheme different from the one in which he is listening. A similar problem is found on a much smaller scale in wireless sensor networks, in which devices are sometimes assigned slots in which to broadcast. Attacking these requires overlapping of not only the channel, but also the broadcast slot and symbol coding. Even for protocols such as IEEE 802.15.4 which support Guaranteed Time Slots (GTS) [6, 194], these are often not implemented as their functionality is unnecessary or requires too high design costs.

On the other hand, injecting from one time-slot to another can be quite handy for increasing the chances of success with a PIP. If the broadcast time is under the control of the attacker, she can slightly advance this time so that the receiver wakes after the true Sync has been transmitted, allowing for reliable PIP reception.

Whitening Some protocols will whiten traffic by combining the frame with a pseudo-random sequence that is either static (in the case of Microsoft 2.4GHz keyboards [10]) or seeded by the low bits of a synchronized clock (as in Bluetooth [11]). Because this is intended to reduce radio interference, rather than to provide cryptographic unpredictability, it poses no trouble if the inner packet's offset from the beginning of a packet is known. Rather, the PIP author can simply perform the inverse of the whitening action, expecting the whitening to transform his message into that which he wishes to place on the air. In Bluetooth, it is furthermore necessary to produce a collision on the six clock bits that seed the pseudo-random number generator.

Symbol Alignment Symbol alignment seems so minor as to avoid mention, except that some protocols use awkward symbol constructions for backward compatibility. As a rule of thumb, data must be aligned to the symbol width. So 2-FSK data needs no alignment, while 4-FSK requires bit-pair alignment. DSSS radios such as 802.15.4 and Wifi require larger symbol sizes, as they sometimes have four or more bits per symbol.

Differential Signaling Some phase-shift keying (PSK) protocols use differential signaling, in which symbols are encoded by the change in phase, rather than by the absolute condition of the phase. Additionally, it is not uncommon for this relationship to change between regions

of a packet. Care must be taken in this case to ensure that the intended phase relationship is constructed within the inner frame.

Inter-frame Gap When injecting multiple PIPs within a single outer-frame, it is necessary to take into account the inter-frame gap requirements imposed both by standards and by their implementations. That is to say, when two frames have no gap between them, the second frame will likely be lost as the receiver is not yet ready for it.

Additionally, as ambiguities in inter-frame gap lengths lead to collisions that damage frame headers, they can be leveraged to produce more reliable PIP injection. On networks with extremely low symbol error rates, such as switched Ethernet, collisions of this sort might be the only way to produce a PIP injection.

Cryptography Cryptography for which the attacker has no key is a much greater impediment to remote PIP injection than it is to a local eavesdropper. This is because the attacker needs to know both the cryptographic key and the counter or nonce. In the case of a known key and nonce, an attacker could conceivably work backward to product a block which, once encrypted, contains a PIP. The exact procedure for doing so depends upon the presence of numerous cryptographic mistakes by the vector protocol and is beyond the scope of this paper.

6 Mitigations

6.1 Reduced Rates and Symbol Sets

802.11b and related protocols change rates and symbol codings within a frame, such that the end of the packet is faster than the preamble [7]. If this were done in reverse, with the rate or symbol set of the body being a subset of that used for the preamble, then the attacker would be unable to create a valid preamble of that protocol. Unfortunately, this solution is impractical both because of the performance hit it will incur and because, as with kernel filtering, the attacker is still able to forge packets of related protocols.

6.2 Cryptography

Cryptography, even otherwise ineffective cryptography, is quite likely the best solution to PIP injection attacks. Encrypting the payload can prevent an attacker from knowing how his bits will appear on the air.

7 Related Work

7.1 Orson Welles

In 1938, Orson Welles and his Mercury Theatre on the Air performed an infamous broadcast of War of the Worlds [12]. The broadcast begins with a brief theatrical introduction, followed by simulated news broadcasts describing an alien invasion of New Jersey that run for *thirty-eight minutes* before the first and only intermission, after which the story shifts to the past tense and a fictional tone. Prior to the intermission, there is not one commercial, not one word out of character, and not one scene in the past tense to clue the listener in on the fictional nature of the broadcast.

While Welles surely was not concerned with attacking digital radios in 1938, his broadcast does follow the general pattern of the attacks in this paper. His PIP in this case is the thirty-eight minute panicked broadcast, while the introduction could be considered an outer header. Listeners who miss the introduction might believe the first act to be factual, just as a digital radio which misses a preamble might interpret the PIP to be a legitimate and full packet.

7.2 Hayes Modems

To switch between data and command mode, many modems use an escape sequence of three bytes, defaulting to "+++". To protect against accidental mode changes resulting from the sequence appearing in data, Hayes Microcomputer Products designed—and received a patent [5] on—the escape sequence of: pause, "+++", pause. When Hayes began to enforce their patent, charging $1 per modem license fees to other manufacturers, some firms simply dropped the mandatory delay in order to avoid patent infringement. This opened up the possibility of a denial-of-service attack, where "+++ATH" would cause the modems to hang-up their connection.

This vulnerability was exploited by Hayes itself in email signatures and press releases [4]. The +++ATH attack offers a classic example of the weakness of in-band signaling.

8 Conceptual impact on network security engineering

Our packet-in-packet technique is made possible by certain features of the affected radio protocols' signaling and can therefore be considered an instance of the diverse class of attacks on in-band signaling. We do not see this problem as just another demonstration of in-band signaling dangers; rather, we see it as an instance of a larger class of attacks with broader implications to the realm of

network security engineering. This larger class consists of attacks against the boundaries of layers, producing unintended cross-layer interactions.

Layering is a seductive abstraction because, almost by definition, layers are easy to conceive of as naturally isolated, with the only data flows between the layers being those provided by the endpoints' network stack APIs. The layer paradigm makes it easy to believe that different designs or implementations of a given layer are interchangeable, and can be largely dealt with (e.g., designed or analyzed for security) independently.

Generally speaking, our attack demonstrates a way of affecting the network stacks' *perception of the lower layer medium and messages* by merely manipulating *the payloads of a higher layer*, notwithstanding the API integrity of either. This shows that natural layer isolation assumptions are a dangerous fallacy. Our technique shows that the behaviors of Physical and Link layers in radio networks are complex enough to warrant reexamining their designs for any such assumptions. As far as we know, analysis of Physical or Link layer designs for cross-layer interaction is not common in industry. This needs to change.

9 Future Work

Although we have only demonstrated our techniques for digital radio, it is quite likely that they are applicable to modem, copper, and fibre optic communications. Examples for each and their likelihood of success would be excellent topics for further research.

IEEE 802.3 frames are defined much like radio frames. Just like a radio transmission, they consist of a Preamble (`aa aa aa aa aa aa aa aa`), a Sync (`ab`), and a Body. While symbol errors are presumably much rarer than in radio protocols, frame collisions can and do occur, particularly when there is a disagreement as to the inter-frame gap. If one Ethernet card begins to transmit before the receiver begins to process new traffic, the receiver will miss the start of a frame, allowing for a PIP to be received as a raw 802.3 frame.

Though we have yet to attempt PIP attacks against Ethernet, the implications could be staggering. While corporate wireless networks are presently protected by cryptography, corporate wired networks are largely unencrypted. An attacker could conceivably inject raw frames either remotely or with reduced privileges.

10 Conclusion

We have demonstrated that in-band signaling mechanisms common to many varieties of digital radio can be abused to inject raw digital frames given control of data in the interior of a frame. We have provided tested examples of using this technique both to escalate between networking layers of a single radio architecture (Section 4.1), to inject raw frames between architectures that differ at the physical layer (Section 4.2), and to evade packet filtering defenses.

Use of in-band signaling to denote the start of a packet is almost universal in digital radios. Our research has demonstrated that an attacker can exploit this property wherever she is able to predict the on-air pattern produced by encapsulated data. This can be mitigated within a networking stack in a number of ways, but only by preventing the attacker from having knowledge of the on-air symbols can cross-stack attacks be effectively prevented.

11 Acknowledgments

This research results in part from a research program at the Institute for Security, Technology, and Society at Dartmouth College, supported by the National Science Foundation under Grant Award Number 1016782. This material is also based in part upon work supported by the Department of Energy under Award Number DE-OE0000097.

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 National Science Foundation. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. 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.

References

[1] ANT+. `http://www.thisisant.com/ant/ant-interoperability`.

[2] J. Cache, H. D. Moore, and skape. Exploiting 802.11 Wireless Driver Vulnerabilities on Windows. *Uninformed.org*, 6, November 2006.

[3] T. Goodspeed. Reversing an RF Clicker.

[4] Hayes Microcomputer Products, Inc. Hayes Press Release: High-Density Rack Communications System, September 1992.

[5] D. A. Heatherington. Modem with improved escape sequence mechanism to prevent escape in response to random occurrence of escape character in transmitted data. Patent 4,549,302, October 1985.

[6] IEEE Computer Society, LAN/MAN Standards Committee, New York, NY. *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 edition, September 2006.

[7] IEEE Computer Society, LAN/MAN Standards Committee, New York, NY. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE 802.11-2007 edition, June 2007.

[8] Jon "Johnny Cache" Ellch and David Maynor. Hijacking a MacBook in 60 seconds. In *BlackHat USA*, August 2006.

[9] LMH. Month of Kernel Bugs (MoKB) archive. `http://projects.info-pull.com/mokb/`, November 2006.

[10] M. Moser and T. Schroeder. Keykeriki 2.0.

[11] D. Spill and A. Bittau. BlueSniff: Eve meets Alice and Bluetooth. In *USENIX Workship on Offensive Technologies*, August 2007.

[12] O. Welles. War of the Worlds. Mercury Theatre on the Air, 1938.