# Radio Exploitation 101

Characterizing, Contextualizing, and Applying Wireless Attack Methods

Matt Knight
Bastille Networks
San Francisco, CA
matt@bastille.net

Marc Newlin
Bastille Networks
Atlanta, GA
marc@bastille.net

*Abstract*—**Reverse engineering wireless physical layers has never been easier, thanks to the commoditization of Software Defined Radio (SDR) technology and an active open source community. However, the successful application of SDR to security challenges requires extensive domain knowledge and insight into radio frequency fundamentals.**

**The goal of this paper, and accompanying presentation, is to highlight how wireless network exploitation is both similar to, and distinct from wired network exploitation, and to offer techniques that will aid security researchers in thinking creatively about wireless reverse engineering and exploit development.**

*Index Terms*—**wireless, security, reverse engineering, software defined radio, radio frequency, internet of things, mobile**

## I. Introduction

The growth of mobile and Internet of Things (IoT) technologies has reshaped the computing landscape as we know it. Wireless networking is a central capability of these developments, and with its advantages comes an expansive and unfamiliar attack surface. While technologies such as 802.11 are common and have been well-exercised, variants such as cellular, low power/ISM standards, and proprietary implementations are rarely fully understood by virtue of the lack of commodity networking interfaces that are required to explore them. *Software Defined Radio* (SDR) has been a major disruptor within the wireless security space, as it generalizes the requisite hardware capabilities to interact with most RF physical layers.

This paper characterizes and defines the types of attacks that are possible within the wireless domain. Comparisons are drawn analogous attacks on wired networks, and special attention is paid to attack models that are unique to RF. Each attack type is paired with a recent real-world example to illustrate each concept. Finally, techniques for defensive mitigation are introduced so that attackers and developers alike can begin to think more strategically when approaching wireless systems.

## II. The Evolution of Network Exploitation

Recent years have seen a flood of novel wireless exploits, with exploitation moving beyond 802.11 and into more obscure standard and proprietary wireless protocols. This can be attributed to the proliferation and commoditization of technologies that provide promiscuous access to the physical layer of the communication stack. For context, we will briefly discuss the evolution of network exploitation.

### A. Network Abstraction Models

Network abstraction models, such as the Open Systems Interconnection (OSI) model, separate communications functions out into generalized components as a means of promoting standardization and interoperability. These abstraction models, however, are arbitrary constructs – that is to say there is no fundamental difference between electrons representing data bits at the Data Link Layer vs. data bits at the Application Layer. However, these abstractions represent boundaries along which vulnerabilities can exist due to imperfect or incomplete integration. [1]

Inspecting data at these boundaries, or on lower layers than manufacturers intended, is a productive means of discovering vulnerabilities. With wireless systems, the lowest layer which manufacturers expose is typically the Data Link Layer (Layer 2) or the Network Layer (Layer 3). Because the radio-based Physical Layer (Layer 1) is implemented in purpose-built silicon, it is often either considered to be out of scope of the security model or taken for granted by system integrators and vendors. However, advances in radio technology has made the inspection of radio-based Physical Layer protocols viable, thus exposing a broad attack surface to hackers.

### B. Commoditization of Early Packet Sniffers

The first commercial network sniffer was released by Network General as the Sniffer Network Analyzer software in 1986. [1] Through the mid-1990s, the software was commonly sold preinstalled on expensive Dolch computers the size of large briefcases. Before long, commodity network cards became capable of integrating with the Sniffer Network Analyzer and applications like it, thus lowering the barrier to inspecting wired network traffic. [2] Today, packet sniffing software like Wireshark and tcpdump makes network analysis easier than ever.

A similar evolution took place with 802.11 in the late 1990s and early 2000s. Monitoring arbitrary 802.11 channels used to be the domain of expensive test equipment. However, with most commercial 802.11 network interfaces now supporting monitor mode, inspecting arbitrary 802.11 traffic can be done with commodity 802.11 chipsets.

### *C. Commodity Software Defined Radio*

Following the adoption of myriad wireless technologies in support of mobile and IoT, we now observe the commoditization of *Software Defined Radio* (SDR). *Software Defined Radio* pushes the architectural hardware/software boundary out closer to the radio, such that generic hardware can implement arbitrary wireless protocols in software. This empowers researchers to interface with any wireless system, as long as they are able to implement the appropriate software. Throughout the 2000s early Software Defined Radios could be had on an academic, government, or military budget; now with commercial products like the *USRP* ($650), *BladeRF* ($420), *HackRF* ($300), and the *RTL-SDR* (~$20), Software Defined Radio is within reach of modestly-funded hackers and independent researchers.

### *D. The Internet of Embedded Systems*

Setting aside marketing buzzwords for a moment, IoT devices, and wireless radios themselves, are *connected embedded systems*. While adding computers to simple machines can lead to increased precision and efficiency, the design and environmental constraints placed on embedded systems make them inherently more vulnerable than traditional platforms.

- **Hardware limitations:** Embedded systems are designed to be small, low-power, cheap, and inexpensive. They often use low-power embedded CPUs with limited computational capacity and radio technologies that trade data rates for endurance. Thus there can be limited computational and networking bandwidth for encryption overhead. Finally, embedded systems sometimes use inexpensive one-time-programmable memory for storing their images, meaning it is not uncommon for systems to be unable to receive software updates once manufactured.
- **Battery powered:** Embedded devices are often battery powered, meaning they need to aggressively duty-cycle to save energy.
- **Limited connectivity:** If connected, embedded systems are connected often using networking technologies that have limited bandwidth and scope. This means it can be difficult to provide sufficient bandwidth for encrypted communication, or to deliver software updates to devices in the field.
- **Complicated deployments:** Embedded systems are often deployed in hard-to-reach places by non-technical installers. They therefore are often required to be simple to install and configure, and either immutable or hard/expensive to reconfigure once deployed. Additionally, embedded systems are often whitelabeled or sold through distributors, making ownership of the software stack a nontrivial matter.
- **High endurance:** Embedded devices are often expected to last for years before replacement.

Given these traits, embedded vulnerabilities can persist for years. Thus, security considerations are essential when weighing the design of embedded systems, and evaluating the security of the systems that connect them to the broader world.

## III. Radio-Based Physical Layers

Wireless communication systems are defined by having a radio-based physical layer (PHY). A radio-based PHY defines how data presented by the Data Link Layer (MAC) gets mapped into electromagnetic phenomena for transmission to a remote receiver. Overall characteristics of the protocol, such as bandwidth, promiscuity, and persistence, can vary based on implementation. All wireless protocols, however, exist within the radio frequency domain; therefore we invoke the following concepts:

- **Radio Spectrum:** Radio waves travel along the electromagnetic spectrum. The radio spectrum can be thought of as a *shared communications bus* which all radio protocols use.
- **Frequency:** Since radio signals are waves, they are periodic and therefore have a frequency. Within our bus-based model, the spectrum is MIMO/multi-input multi-output, with this multiplexing occurring by frequency.
- **Channel:** All radio protocols have some notional implementation of a *channel*. The channel is characterized by the amount of *bandwidth* the protocol utilizes, centered about the *center frequency* of the signal. There may be one or several center frequencies depending on whether the protocol channel hops or not. Channels may overlap, and transmissions may collide – this is a reality of working within a shared medium. Traditional electrical data buses, such as SPI or CAN, are coordinated or use deconfliction techniques to avoid collisions; wireless protocols use channel monitoring and retransmissions to mitigate the impact of collisions.
- **Signal Power and Noise:** Radio waves propagate in a similar manner to audio waves. Both gradually lose power as they radiate away from their source, until they are eventually lost beneath the *noise floor* – that is, they lose power to the point where they become

not discernible from the background noise. The noise floor is influenced by both the radio receiver itself and environmental conditions, including intentional and unintentional (interfering) radio emissions.

## IV. Radio Exploitation 101

Here we begin to outline our wireless threat taxonomy, with particular emphasis on what makes wireless exploitation and defense distinct from the same on wired networks. To this end, we have consolidated noteworthy techniques into the following attack models. This non-exhaustive list includes *sniffing*, *wardriving*, *replay attacks*, *jamming*, *MAC-layer channel reservation abuse*, *evil twin attacks*, *firmware update exploitation*, and *physical layer protocol abuse*. For each type of attack we will describe:

- **Method of attack:** In plain English, how is this attack performed?
- **Potential impact:** What are the consequences for the victims of such an attack?
- **Analogous attack on wired networks:** Is there an analogous attack on wired networks? If not, how and why is this attack scenario unique to RF?
- **A recent example of such an attack:** To provide context, what is a real-world example of a system or organization that has fallen victim to such an attack?
- **Limitations and defensive mitigations:** What sets of circumstances have to align to facilitate this attack? How broadly viable is it? What steps can defenders take to mitigate their exposure?
- **Description of our paired Hack in the Box GSEC demo:** If there is a live demo from the associated ”Radio Exploitation 101” Hack in the Box talk, it will be explained here.

### A. Sniffing

We begin with *sniffing*, the passive observation of wireless network traffic. *Sniffing* is noteworthy because the wireless domain enables truly promiscuous sniffing with no direct physical access.

- **Method of attack:** *Sniffing* is performed by using a radio receiver to passively receive wireless network traffic.
- **Potential impact:** Data loss, device/network discovery
- **Wired analogue:** Sniffing exists in wired contexts too. However, wired network sniffing requires direct physical access to a network or bus – in other words, one must be physically connected to the network in order to observe its traffic. Since electromagnetic signals by nature radiate throughout free space, listeners other than the intended receiver can remotely monitor network traffic without detection.
- **Recent example:** Marc Newlin’s 2016 Mousejack vulnerability revealed that many wireless keyboards failed to properly encrypt their keystrokes, with many vendors forgoing encryption entirely – thus, sniffing their traffic was trivial.
- **Limitations and defensive mitigations:** While attackers do not require direct access to a bus, sniffing still requires a degree of physical proximity to the transmitter. Additionally, attackers must possess a radio that is compatible with the protocol to be sniffed. Defenders can mitigate their exposure by encrypting traffic so that sniffed packets will be of limited utility to attackers.
- **Hack in the Box GSEC demo:** Sniffing keystrokes from an unencrypted wireless keyboard.

### B. Wardriving

*Wardriving* is a type of *sniffing* that refers to the act of searching for wireless networks or devices. Its name originates from 802.11 wardriving, where 802.11 access points are sought using equipment within a moving vehicle. With the growth of mobile and IoT protocols, wardriving now refers to discovering non-802.11 RF networks as well.

- **Method of attack:** *Wardriving* can be passive or active. Passive scenarios involve the attacker *sniffing* on channels of interest, looking for wireless traffic that denotes the presence of a network or device(s). Active scenarios involve the attacker transmitting messages intended to induce a response from present devices or infrastructure, and then *sniffing* for said responses.
- **Potential impact:** Discovery of devices and networks, identifying exploitable devices
- **Wired analogue:** Active wardriving is analogous to port scanning. Just as port scanning is a way of discovering potentially exploitable services running on an endpoint, active wardriving is a means of discovering and enumerating potentially exploitable devices within an environment. Additionally, just as the *nmap* port scanning tool provides operating system fingerprinting through wired querying, *wardriving* enables device fingerprinting through wireless characteristics.
- **Recent example:** Wardriving for 802.15.4 networks is built in to the *Killerbee* 802.15.4/ZigBee attack framework. The *zbstumbler* script hops from channel to channel, sending broadcast beacons and listening for beacon responses from network coordinators.
- **Limitations and defensive mitigations:** As with *sniffing*, attackers must have a degree of physical proximity to the sought after wireless devices in order to wardrive for them. This can be overcome through the use of directional equipment, as was done with the proliferation of jury-rigged *cantennas* during the peak of 802.11 wardriving. Wardriving is a conspicuous process, so defenders can mitigate exposure by being aware of its signatures – for instance, seeing an atypical flood of probe or beacon requests across consecutive channels.

- **Hack in the Box GSEC demo:** Beaconing for 802.15.4 coordinators.

## C. Replay Attack

*Replay attacks* involve retransmitting a previously captured transmission, possibly to induce a previously observed state change or action within the network. The *replay attack* may involve retransmitting a captured raw PHY-layer payload or the synthesis of a new frame based on decoded data.

- **Method of attack:** An attacker must first capture transmission of interest that is correlated with the action on the target they wish to induce – that is, the transmission they intend to replay. This can either be a raw IQ spectrum capture or a decoded packet payload. Software Defined Radio can produce either as long as it has the appropriate decoding stack behind it. The captured transmission can then be replayed to induce the intended action on the network, either by replaying the raw IQ through the appropriate Software Defined Radio or by generating a new transmission from the decoded packet payload.
- **Potential impact:** Change the state of a network, or induce a behavior by a device on a network
- **Wired analogue:** Replay attacks exist in wired contexts.
- **Recent example:** The April 2017 Dallas tornado emergency alert siren attack is widely believed to be an RF replay attack. The attacker likely used a Software Defined Radio to capture the unencrypted, unauthenticated wireless signal used to test the sirens and replay said signal at a later date. [3]
- **Limitations and defensive mitigations:** Replay attacks can be defeated by enforcing cryptographic authentication and freshness. Cryptographic authentication is the practice of using cryptography to establish trust among two or more endpoints – essentially using cryptography to sign messages as a means of validating their authenticity. Freshness refers to tracking a sequence number within a message frame – freshness is not a security feature in and of itself, but when combined with authentication and encryption can make replay attacks far harder to execute.
- **Hack in the Box GSEC demo:** Replay attack against a Fortress home security smart home alarm device and SOS siren (similar scenario to the Dallas tornado siren).

## D. Jamming

*Jamming* is perhaps the most well-known attack on wireless systems. Since the radio frequency domain can be thought of as the bus that all wireless systems share, loading it up with powerful wideband noise or spurious traffic is an effective way of denying a communications channel.

- **Method of attack:** *Jamming* in its simplest form can be conducted by transmitting noise within the target network's RF channel – that is, at the same frequency and with sufficient bandwidth and power. In lieu of wideband noise, rapidly sending arbitrary packets while ignoring channel contention can have the same effect.
- **Potential impact:** Denies legitimate network traffic, disrupts network state
- **Wired analogue:** Wired Denial of Service attacks are analogous to jamming – that is, filling a communications channel with spurious traffic intended to overload a communications channel.
- **Recent example:** In 2014, researcher Logan Lamb discovered that wireless home security system sensors could be thwarted by using a wideband jammer to block signals traveling from the sensors to the control panel. [4]
- **Limitations and defensive mitigations:** From the attacker's perspective, jamming is self-defeating – jamming denies the target network, but also the attacker's ability to monitor attempted transmissions on the jammed network. It can also be fairly easy to detect. Defenders can mitigate jamming with the implementation of jam detection mechanisms. These can be practically implemented on embedded devices by polling the clear-channel assessment (CCA) mechanism available on many hardware radios or by taking a power measurement of the channel. Short of this, devices can use network health diagnostics and statistics to determine when their network may be under attack. Examples on how to evade these jam detection mechanisms are outlined below.
- **Hack in the Box GSEC demo:** Logan Lamb's alarm system jammer.

*Evasive Jamming:* As mentioned, device manufacturers can take a defense posture against jamming by implementing jam detection countermeasures on their devices. Here are two ways that that clever attackers may be able to circumvent such jam detection mechanisms:

- **Duty Cycled Jamming:** Duty-cycling a jammer, or pulsing it on and off, is one way of defeating CCA-based jam detection mechanisms. If done at an appropriate rate, cycling the jammer and allowing the channel to appear open from the perspective of the device under attack will keep the detection functions from triggering while still denying the channel.
- **Reflexive Jamming:** Reflexive jamming is one way of denying wireless communications that is more difficult to detect. Reflexive jamming involves having the radio doing the jamming to wait for a transmission to begin, jamming briefly only once a transmission is detected. The jamming signal collides with part of the packet, corrupting some of its symbols and causing it to fail the receiver's expected CRC check. Thus, the jammer denies the channel, or even a specific device/set of devices, while remaining active for the shortest possible time.

## E. Data Link Layer Channel Reservation Abuse

The 802.11 data link layer employs *carrier-sense multiple access with collision avoidance* (CSMA/CA), an algorithm which prevents two nodes from transmitting simultaneously. CSMA/CA uses two methods to determine if a channel is currently in use by another node: *carrier-sensing*, and *virtual carrier-sensing*. *Carrier-sensing* detects a busy channel by measuring RF energy. The CSMA/CA state machine assumes that another node is currently transmitting when it detects RF energy above a predefined threshold, and waits for the channel to return to idle before transmitting. *Virtual carrier-sensing* acts on the *frame duration field* in the 802.11 MAC header, which defines the expected time, in microseconds, required to transmit the packet and receive an ACK. In order to save power, when an 802.11 node receives a packet, it will assume the channel is occupied for the duration specified in the MAC header.

- **Method of attack:** Virtual carrier-sensing can be abused to effectively jam an 802.11 channel without needing to transmit at a high duty cycle. By transmitting a frame with an empty payload, and a frame duration value of 32,767 or greater, other nodes in range will remain inactive for 32ms. Therefore, an 802.11 channel can be effectively jammed by transmitting 30 such frames per second.
- **Potential impact:** Denial of service: legitimate 802.11 nodes are denied access to the RF channel.
- **Wired analogue:** Denial of service, however this CSMA/CA virtual carrier-sensing is unique to wireless networking protocols.
- **Recent example:** There are no known recent examples of this attack. Bastian Bloessl is credited with suggesting the attack vector.
- **Limitations and defensive mitigations:** The network allocation vector (NAV) counter, which stores the remaining duration for which an 802.11 channel is expected to be occupied, has a maximum value of 32,767. This limits the the effect of a single malicious packet to 32ms, requiring the attacker to continually transmit, albeit with a low duty cycle.
- **Hack in the Box GSEC demo:** 802.11 virtual carrier-sense abuse attack.

## F. *Evil Twin*

An *Evil Twin* attack involves standing up a decoy device or rogue access point that mimics trusted infrastructure, such that it tricks victims into connecting to it. It is a way of automating the establishment of trust to eavesdrop on or interact with clients. The classic example of an 802.11 *Evil Twin* is the *Wi-Fi Pineapple Karma attack* [5], however examples exist on other protocols as well.

- **Method of attack:** The attacker must first capture the defining metadata of the infrastructure to be mimicked. This may include RF channel information and addressing information, for example MAC addresses and SSIDs for 802.11. These metadata will differ for other protocols. Then a decoy device can be set up using this extracted configuration. If done convincingly, clients may elect to trust the mimic and connect to it instead of the legitimate AP. This technique can be combined with other techniques such as jamming to further deny the legitimate infrastructure and make the mimic appear even more desirable.
- **Potential impact:** Eavesdropping and tampering with network traffic, data loss, tracking devices
- **Wired analogue:** Wireless *Evil Twin* attacks are analogous to ARP cache poisoning, or ARP spoofing. With ARP poisoning, an attacker injects fraudulent ARP responses into a local area network to get targeted clients to associate the attacker's MAC address with a target IP address. This results in the targeted clients routing messages to the attacker rather than the intended recipient. *Evil Twin* attacks have similar characteristics – the rogue device hijacks routing as a means of intercepting traffic.
- **Recent example:** The *Wi-Fi Pineapple* is a well-known device capable of executing *Evil Twin* attacks. Additionally, IMSI Catchers such as the Stingray are Evil Twins – rogue cell towers such as these exploit the lack of mutual authentication in the GSM cellular protocol to masquerade as legitimate cellular infrastructure.
- **Limitations and defensive mitigations:** Because of the complexity associated with convincingly mimicking an existing device or communications protocol, Evil Twin attacks can be difficult to execute. In addition, if the attacker's rogue device has to compete side-by-side with legitimate device, it may also be necessary for the attacker to deny the legitimate device with other offensive techniques such as jamming.
  From the defender's perspective, *Evil Twin* attacks can be mitigated through the proper use of *cryptographic mutual authentication*. Authentication allows the devices to trust the identity of a recipient before electing to associate with or route traffic to them. GSM IMSI catchers present an example of mutual authentication *not* being used properly. Within the GSM protocol, basestations can authenticate the identity of handsets, however handsets *do not* authenticate basestations. Thus, it is trivial for an attacker to stand up a rogue basestation, because handsets have no robust way of verifying their identity.
- **Hack in the Box GSEC demo:** Intercepting a cell phone with a rogue GSM basestation.

## G. *Firmware Update Mechanisms*

Attacks on *wireless firmware update mechanisms* enable attackers to execute arbitrary software and gain persistence on a device.

- **Method of attack:** A highly generalized case of this attack involves the attacker identifying the presence of a wireless firmware update mechanism within a target device, preparing a modified binary, overcoming firmware encryption or secure boot (if present), and delivering the

modified binary to the target device. Such a modified binary could implement arbitrary malicious features, including but not limited to self-propagating to other similar devices as a worm, or exfiltrating network data back to the attacker, or bricking the device.
- **Potential impact:** Attacker gaining persistence on the device, self-propagation (i.e. worm), denial of service, data loss
- **Wired analogue:** Attacking wireless firmware update mechanisms presents opportunities to exploit embedded devices in manners similar to how malware operates on traditional endpoints. Additionally, the ability to have infected devices infect other devices is directly comparable to traditional worms.
- **Recent example:** Eyal Ronen, Colin O'Flynn, Adi Shamir, and Achi-Or Weingarten's "ZigBee Chain Reaction" from 2016 is a highly publicized recent example of an attack on a wireless firmware update mechanism. Their attack showcased deploying malicious firmware to a set of Philips Hue lightbulbs from a radio mounted on a drone. In addition to having to figure out ZigBee's wireless firmware update mechanics, they also had to obtain the firmware signing key used by Philips to craft their secure boot images. [6]
- **Limitations and defensive mitigations:** Defenders can mitigate attacks like this by implementing modern best practices such as secure boot/firmware encryption/image signing and network encryption.
- **Hack in the Box GSEC demo:** No demo, but a discussion of the aforementioned ZigBee firmware OTA attack.

### H. Physical Layer Protocol Abuse

*Physical Layer Protocol Abuse* attacks, as we style them, involve sending wireless transmissions that exploit irregularities and corner cases in the receiver's physical layer state machine. Effects of these attacks can vary, from being able to transmit from a device without direct control of the radio to being able to send hidden or difficult-to-detect messages within the radio spectrum.

- **Method of attack:** The techniques used to implement these attacks, and even their desired outcomes, are varied in nature, so generalizing them is difficult. One example involves exploiting the structure of the physical layer frame by embedding the symbols that comprise an entire PHY frame within the payload of another packet, such that two well-formed packets are sent with a single call to the radio. Other examples include using illegal preamble and header values to create transmissions that certain radio state machines may not be able to receive, or exploiting symbol mapping tables to exploit corner cases in a receiver's state machine.
- **Potential impact:** Wireless IDS evasion, device fingerprinting, privilege escalation
- **Wired analogue:** Covert messaging has been demonstrated on the 802.3 Ethernet physical layer.
- **Recent example:** Perhaps the best-known example of physical layer abuse is the *packet-in-packet* 802.15.4 attack from 2011. [7] The same group from Dartmouth also developed series of 802.15.4 selective evasion techniques by wirelessly fingerprinting common chipsets. [8]
- **Limitations and defensive mitigations:** Since defenders have to assign significant trust to their hardware, there are few practical options for protecting against creative wireless physical layer attacks.
- **Hack in the Box GSEC demo:** 802.15.4 physical layer evasion techniques.

## V. Conclusions

As embedded systems continue to permeate our digital lives, wireless communication systems are slated to become even more ubiquitous and diverse. As these systems continue to assume critical functions within society, it is paramount that device manufacturers and integrators consider the novel challenges that accompany such interfaces. Rather than dismissing RF as voodoo that magically makes bits appear on faraway devices, remember that radios are deterministic state machines with behavior that can be rationally understood. Finally, remember that radios are hardware and hardware makes vulnerabilities last: thus, grasping wireless today is an essential step to securing communications for years to come.

## VI. Further Reading

For further reading, consider consulting the following materials:

### A. RF Physical Layer Fundamentals and Reverse Engineering Techniques: "So You Want to Hack Radios" Series

- **Shmoocon:** https://www.youtube.com/watch?v=L3udJnRe4vc
- **Troopers17:** https://www.youtube.com/watch?v=OFRwqpH9zAQ
- **Hack in the Box Amsterdam 2017:** https://www.youtube.com/watch?v=QeoGQwT0Z1Y

### B. Protocol Deep Dives

- **LoRa:** Matt's presentation on the LoRa physical layer from 33c3: https://media.ccc.de/v/33c3-7945-decoding_the_lora_phy
- **Mousejack:** Marc's presentation on vulnerabilities in the nRF24 wireless keyboard and mouse protocol from DEF CON 24: https://www.youtube.com/watch?v=00A36VABIA4

### *C. Applying Open Source Intelligence (OSINT) to the Reverse Engineering Process*

- **Hack in the Box Amsterdam 2016:** https://www.youtube.com/watch?v=JUAiav674D8

## References

[1] A. Joch, "Network sniffers," Article, 2001, http://www.computerworld.com/article/2583125/lan-wan/network-sniffers.html.

[2] "Network general analyzer sniffs out network trouble," Magazine Article, 1996, https://books.google.com/books?id=Ij0EAAAAMBAJ&lpg=PA47-IA6&ots=CKwhmoptiu&dq=dolch%20sniffer%20history&pg=PA47-IA6#v=onepage&q=dolch%20sniffer%20history&f=false.

[3] B. Seeber, "Dallas siren attack," Online Case Study, 2017, https://www.bastille.net/blogs/2017/4/17/dallas-siren-attack.

[4] L. Lamb, "Home insecurity: No alarms, false alarms, and sigint," Paper, 2017, https://media.defcon.org/DEF%20CON%2022/DEF%20CON%2022%20presentations/Logan%20Lamb/DEFCON-22-Logan-Lamb-HOME-INSECURITY-NO-ALARMS-FALSE-ALARMS-AND-SIGINT-WP.pdf.

[5] S. Helme, "The wifi pineapple - using karma and dnsspoof to snag unsuspecting victims," Blog Post, 2013, https://scotthelme.co.uk/wifi-pineapple-karma-dnsspoof/.

[6] E. Ronen, C. O'Flynn, A. Shamir, and A.-O. Weingarten, "Iot goes nuclear: Creating a zigbee chain reaction," Paper, 2016, http://iotworm.eyalro.net/iotworm.pdf.

[7] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers, "Packets in packets: Orson welles? in-band signaling attacks for modern radios," Paper, 2011, https://www.usenix.org/legacy/event/woot11/tech/final_files/Goodspeed.pdf.

[8] I. R. Jenkins, R. Shapiro, S. Bratus, T. Goodspeed, R. Speers, and D. Dowd, "Speaking the local dialect: Exploiting differences between ieee 802.15.4 receivers with commodity radios for fingerprinting, targeted attacks, and wids evasion," Paper, 2014, http://www.cs.dartmouth.edu/reports/TR2014-749.pdf.